

# Quadratic Sorting Algorithms

## Using C++ Sorting

sorting a vector of  
pairs  
setting up C++ code

## Selection Sort

the selection sort  
algorithm  
C++ code for  
selection sort

## Bubble Sort

the bubble sort  
algorithm  
C++ code for bubble  
sort

## Insertion Sort

the insertion sort  
algorithm  
C++ code for  
insertion sort

- 1 **Using C++ Sorting**  
sorting a vector of pairs  
setting up C++ code
- 2 **Selection Sort**  
the selection sort algorithm  
C++ code for selection sort
- 3 **Bubble Sort**  
the bubble sort algorithm  
C++ code for bubble sort
- 4 **Insertion Sort**  
the insertion sort algorithm  
C++ code for insertion sort

# Quadratic Sorting Algorithms

## Using C++ Sorting

sorting a vector of  
pairs

setting up C++ code

## Selection Sort

the selection sort  
algorithm

C++ code for  
selection sort

## Bubble Sort

the bubble sort  
algorithm

C++ code for bubble  
sort

## Insertion Sort

the insertion sort  
algorithm

C++ code for  
insertion sort

- 1 Using C++ Sorting  
    sorting a vector of pairs  
    setting up C++ code
- 2 Selection Sort  
    the selection sort algorithm  
    C++ code for selection sort
- 3 Bubble Sort  
    the bubble sort algorithm  
    C++ code for bubble sort
- 4 Insertion Sort  
    the insertion sort algorithm  
    C++ code for insertion sort

# sorting a vector of pairs

## Using C++ Sorting

sorting a vector of  
pairs

setting up C++ code

## Selection Sort

the selection sort  
algorithm

C++ code for  
selection sort

## Bubble Sort

the bubble sort  
algorithm

C++ code for bubble  
sort

## Insertion Sort

the insertion sort  
algorithm

C++ code for  
insertion sort

Consider `vector< pair<int, char> > v` as a frequency table. We sort on the `int` key.

5 random items : (82,i)(90,d)(42,x)(54,r)(31,z)

We sort in increasing order of keys:

- 1 customized compare function,
- 2 illustrate stable sort.

Equal keys retain their relative order in a *stable* sort.

For example: (23,c) .. (23,a)  $\Rightarrow$  (23,c) (23,a)

We assume we sort container with *random-access iterator*.

For convenience: use subscripting operator [ ] on vectors.

# sorting a vector of pairs

## Using C++ Sorting

sorting a vector of  
pairs

setting up C++ code

## Selection Sort

the selection sort  
algorithm

C++ code for  
selection sort

## Bubble Sort

the bubble sort  
algorithm

C++ code for bubble  
sort

## Insertion Sort

the insertion sort  
algorithm

C++ code for  
insertion sort

Consider `vector< pair<int, char> > v` as a frequency table. We sort on the `int` key.

5 random items : (82,i)(90,d)(42,x)(54,r)(31,z)

We sort in increasing order of keys:

- 1 customized compare function,
- 2 illustrate stable sort.

Equal keys retain their relative order in a *stable* sort.

For example: (23,c) .. (23,a)  $\Rightarrow$  (23,c) (23,a)

We assume we sort container with *random-access iterator*.  
For convenience: use subscripting operator [ ] on vectors.

# sorting a vector of pairs

## Using C++ Sorting

sorting a vector of  
pairs

setting up C++ code

## Selection Sort

the selection sort  
algorithm

C++ code for  
selection sort

## Bubble Sort

the bubble sort  
algorithm

C++ code for bubble  
sort

## Insertion Sort

the insertion sort  
algorithm

C++ code for  
insertion sort

Consider `vector< pair<int, char> > v` as a frequency table. We sort on the `int` key.

5 random items : (82,i)(90,d)(42,x)(54,r)(31,z)

We sort in increasing order of keys:

- 1 customized compare function,
- 2 illustrate stable sort.

Equal keys retain their relative order in a *stable* sort.

For example: (23,c) .. (23,a)  $\Rightarrow$  (23,c) (23,a)

We assume we sort container with *random-access iterator*.  
For convenience: use subscripting operator [ ] on vectors.

Using C++  
Sortingsorting a vector of  
pairs

setting up C++ code

## Selection Sort

the selection sort  
algorithmC++ code for  
selection sort

## Bubble Sort

the bubble sort  
algorithmC++ code for bubble  
sort

## Insertion Sort

the insertion sort  
algorithmC++ code for  
insertion sort

# Motivation

why bother with quadratic sorts?

After STL sort and stable sort,  
we study selection, bubble, and insertion sort.

The algorithms are quadratic cost sorting algorithms,  
i.e.: for sequences of size  $n$ , their cost is  $O(n^2)$ .

Why bother with these less efficient algorithms?

- 1 these algorithms are straightforward
- 2 for moderate dimensions, quadratic cost is not so bad
- 3 understand the basis for advanced sorting algorithms

Using C++  
Sortingsorting a vector of  
pairs

setting up C++ code

## Selection Sort

the selection sort  
algorithmC++ code for  
selection sort

## Bubble Sort

the bubble sort  
algorithmC++ code for bubble  
sort

## Insertion Sort

the insertion sort  
algorithmC++ code for  
insertion sort

# Motivation

why bother with quadratic sorts?

After STL sort and stable sort,  
we study selection, bubble, and insertion sort.

The algorithms are quadratic cost sorting algorithms,  
i.e.: for sequences of size  $n$ , their cost is  $O(n^2)$ .

Why bother with these less efficient algorithms?

- 1 these algorithms are straightforward
- 2 for moderate dimensions, quadratic cost is not so bad
- 3 understand the basis for advanced sorting algorithms

# Quadratic Sorting Algorithms

## Using C++ Sorting

sorting a vector of  
pairs

setting up C++ code

## Selection Sort

the selection sort  
algorithm

C++ code for  
selection sort

## Bubble Sort

the bubble sort  
algorithm

C++ code for bubble  
sort

## Insertion Sort

the insertion sort  
algorithm

C++ code for  
insertion sort

- 1 Using C++ Sorting  
sorting a vector of pairs  
setting up C++ code
- 2 Selection Sort  
the selection sort algorithm  
C++ code for selection sort
- 3 Bubble Sort  
the bubble sort algorithm  
C++ code for bubble sort
- 4 Insertion Sort  
the insertion sort algorithm  
C++ code for insertion sort

## a random vector

Using C++  
Sorting

sorting a vector of  
pairs

setting up C++ code

## Selection Sort

the selection sort  
algorithm

C++ code for  
selection sort

## Bubble Sort

the bubble sort  
algorithm

C++ code for bubble  
sort

## Insertion Sort

the insertion sort  
algorithm

C++ code for  
insertion sort

```
vector< pair<int,char> >
    random_vector ( int n )
{
    vector< pair<int,char> > v;

    for(int i=0; i<n; i++)
    {
        pair<int,char> p;
        p.first = 10 + (rand() % 90);
        p.second = 'a' + rand() % 26;
        v.push_back(p);
    }

    return v;
}
```

## a random vector

Using C++  
Sorting

sorting a vector of  
pairs

setting up C++ code

## Selection Sort

the selection sort  
algorithm

C++ code for  
selection sort

## Bubble Sort

the bubble sort  
algorithm

C++ code for bubble  
sort

## Insertion Sort

the insertion sort  
algorithm

C++ code for  
insertion sort

```
vector< pair<int,char> >  
  random_vector ( int n )  
{  
  vector< pair<int,char> > v;  
  
  for(int i=0; i<n; i++)  
  {  
    pair<int,char> p;  
    p.first = 10 + (rand() % 90);  
    p.second = 'a' + rand() % 26;  
    v.push_back(p);  
  }  
  
  return v;  
}
```

# write and comparison

## Using C++ Sorting

sorting a vector of  
pairs

setting up C++ code

## Selection Sort

the selection sort  
algorithm

C++ code for  
selection sort

## Bubble Sort

the bubble sort  
algorithm

C++ code for bubble  
sort

## Insertion Sort

the insertion sort  
algorithm

C++ code for  
insertion sort

```
void write_vector ( vector< pair<int,char> > v )
{
    for(int i=0; i<v.size(); i++)
        cout << "(" << v[i].first
            << "," << v[i].second
            << ")";
}

struct less_than // defines "<"
{
    bool operator()(const pair<int,char>& a,
                    const pair<int,char>& b)
    {
        return (a.first < b.first);
    }
};
```

## write and comparison

Using C++  
Sorting

sorting a vector of  
pairs

setting up C++ code

## Selection Sort

the selection sort  
algorithm

C++ code for  
selection sort

## Bubble Sort

the bubble sort  
algorithm

C++ code for bubble  
sort

## Insertion Sort

the insertion sort  
algorithm

C++ code for  
insertion sort

```
void write_vector ( vector< pair<int,char> > v )
{
    for(int i=0; i<v.size(); i++)
        cout << "(" << v[i].first
            << "," << v[i].second
            << ")";
}

struct less_than // defines "<"
{
    bool operator()(const pair<int,char>& a,
                    const pair<int,char>& b)
    {
        return (a.first < b.first);
    }
};
```

## the main program

Using C++  
Sorting

sorting a vector of  
pairs

setting up C++ code

## Selection Sort

the selection sort  
algorithm

C++ code for  
selection sort

## Bubble Sort

the bubble sort  
algorithm

C++ code for bubble  
sort

## Insertion Sort

the insertion sort  
algorithm

C++ code for  
insertion sort

```
int main()
{
    cout << "give n : ";
    int n; cin >> n;
    srand(time(0));
    vector< pair<int,char> > v = random_vector(n);
    cout << n << " random items : ";
    write_vector(v); cout << endl;

    cout << "stable sort ? (y/n) ";
    char ans; cin >> ans;
    if(ans == 'y')
        stable_sort(v.begin(),v.end(),less_than());
    else
        sort(v.begin(),v.end(),less_than());

    cout << "the sorted vector : ";
    write_vector(v); cout << endl;
}
```

## the main program

Using C++  
Sorting

sorting a vector of  
pairs

setting up C++ code

## Selection Sort

the selection sort  
algorithm

C++ code for  
selection sort

## Bubble Sort

the bubble sort  
algorithm

C++ code for bubble  
sort

## Insertion Sort

the insertion sort  
algorithm

C++ code for  
insertion sort

```
int main()
{
    cout << "give n : ";
    int n; cin >> n;
    srand(time(0));
    vector< pair<int,char> > v = random_vector(n);
    cout << n << " random items : ";
    write_vector(v); cout << endl;

    cout << "stable sort ? (y/n) ";
    char ans; cin >> ans;
    if(ans == 'y')
        stable_sort(v.begin(),v.end(),less_than());
    else
        sort(v.begin(),v.end(),less_than());

    cout << "the sorted vector : ";
    write_vector(v); cout << endl;
```

## the main program

Using C++  
Sorting

sorting a vector of  
pairs

setting up C++ code

## Selection Sort

the selection sort  
algorithm

C++ code for  
selection sort

## Bubble Sort

the bubble sort  
algorithm

C++ code for bubble  
sort

## Insertion Sort

the insertion sort  
algorithm

C++ code for  
insertion sort

```
int main()
{
    cout << "give n : ";
    int n; cin >> n;
    srand(time(0));
    vector< pair<int,char> > v = random_vector(n);
    cout << n << " random items : ";
    write_vector(v); cout << endl;

    cout << "stable sort ? (y/n) ";
    char ans; cin >> ans;
    if(ans == 'y')
        stable_sort(v.begin(),v.end(),less_than());
    else
        sort(v.begin(),v.end(),less_than());

    cout << "the sorted vector : ";
    write_vector(v); cout << endl;
```

# Quadratic Sorting Algorithms

## Using C++ Sorting

sorting a vector of  
pairs  
setting up C++ code

## Selection Sort

the selection sort  
algorithm

C++ code for  
selection sort

## Bubble Sort

the bubble sort  
algorithm

C++ code for bubble  
sort

## Insertion Sort

the insertion sort  
algorithm

C++ code for  
insertion sort

- 1 Using C++ Sorting  
sorting a vector of pairs  
setting up C++ code
- 2 Selection Sort  
the selection sort algorithm  
C++ code for selection sort
- 3 Bubble Sort  
the bubble sort algorithm  
C++ code for bubble sort
- 4 Insertion Sort  
the insertion sort algorithm  
C++ code for insertion sort

# Selection Sort

## Using C++ Sorting

sorting a vector of  
pairs

setting up C++ code

## Selection Sort

the selection sort  
algorithm

C++ code for  
selection sort

## Bubble Sort

the bubble sort  
algorithm

C++ code for bubble  
sort

## Insertion Sort

the insertion sort  
algorithm

C++ code for  
insertion sort

Sort a sequence of  $n$  elements:

- 1 get the minimum of the sequence,
- 2 place the minimum in front of the sequence,
- 3 sort the remaining  $n - 1$  elements.

With an index  $i$  we keep track of the elements in the sequence still to be sorted.

For  $i$  from 0 to  $n - 1$ :

- 1 select the minimum  $m$  in  $v_i, v_{i+1}, \dots, v_n$ ;
- 2 if  $m \neq v_i$  then swap  $m \leftrightarrow v_i$ .

Loop invariant: first  $i$  elements are smallest.

# Selection Sort

## Using C++ Sorting

sorting a vector of  
pairs  
setting up C++ code

## Selection Sort

the selection sort  
algorithm

C++ code for  
selection sort

## Bubble Sort

the bubble sort  
algorithm

C++ code for bubble  
sort

## Insertion Sort

the insertion sort  
algorithm

C++ code for  
insertion sort

Sort a sequence of  $n$  elements:

- 1 get the minimum of the sequence,
- 2 place the minimum in front of the sequence,
- 3 sort the remaining  $n - 1$  elements.

With an index  $i$  we keep track of the elements in the sequence still to be sorted.

For  $i$  from 0 to  $n - 1$ :

- 1 select the minimum  $m$  in  $v_i, v_{i+1}, \dots, v_n$ ;
- 2 if  $m \neq v_i$  then swap  $m \leftrightarrow v_i$ .

Loop invariant: first  $i$  elements are smallest.

# Selection Sort

## Using C++ Sorting

sorting a vector of  
pairs

setting up C++ code

## Selection Sort

the selection sort  
algorithm

C++ code for  
selection sort

## Bubble Sort

the bubble sort  
algorithm

C++ code for bubble  
sort

## Insertion Sort

the insertion sort  
algorithm

C++ code for  
insertion sort

Sort a sequence of  $n$  elements:

- 1 get the minimum of the sequence,
- 2 place the minimum in front of the sequence,
- 3 sort the remaining  $n - 1$  elements.

With an index  $i$  we keep track of the elements in the sequence still to be sorted.

For  $i$  from 0 to  $n - 1$ :

- 1 select the minimum  $m$  in  $v_i, v_{i+1}, \dots, v_n$ ;
- 2 if  $m \neq v_i$  then swap  $m \leftrightarrow v_i$ .

Loop invariant: first  $i$  elements are smallest.

# Selection Sort

## Using C++ Sorting

sorting a vector of  
pairs

setting up C++ code

## Selection Sort

the selection sort  
algorithm

C++ code for  
selection sort

## Bubble Sort

the bubble sort  
algorithm

C++ code for bubble  
sort

## Insertion Sort

the insertion sort  
algorithm

C++ code for  
insertion sort

Sort a sequence of  $n$  elements:

- 1 get the minimum of the sequence,
- 2 place the minimum in front of the sequence,
- 3 sort the remaining  $n - 1$  elements.

With an index  $i$  we keep track of the elements in the sequence still to be sorted.

For  $i$  from 0 to  $n - 1$ :

- 1 select the minimum  $m$  in  $v_i, v_{i+1}, \dots, v_n$ ;
- 2 if  $m \neq v_i$  then swap  $m \leftrightarrow v_i$ .

Loop invariant: first  $i$  elements are smallest.

1 Nov 2010

## running an example

Using C++  
Sorting

sorting a vector of  
pairs  
setting up C++ code

Selection Sort  
the selection sort  
algorithm

C++ code for  
selection sort

## Bubble Sort

the bubble sort  
algorithm  
C++ code for bubble  
sort

Insertion Sort  
the insertion sort  
algorithm

C++ code for  
insertion sort

```
5 random items : (24,b)(96,k)(12,t)(99,g)(26,w)
```

```
after 2 <-> 0 : (12,b)(96,k)(24,t)(99,g)(26,w)
```

```
after 2 <-> 1 : (12,b)(24,k)(96,t)(99,g)(26,w)
```

```
after 4 <-> 2 : (12,b)(24,k)(26,t)(99,g)(96,w)
```

```
after 4 <-> 3 : (12,b)(24,k)(26,t)(96,g)(99,w)
```

```
the sorted vector : (12,b)(24,k)(26,t)(96,g)(99,w)
```

1 Nov 2010

## running an example

Using C++  
Sorting

sorting a vector of  
pairs  
setting up C++ code

Selection Sort  
the selection sort  
algorithm

C++ code for  
selection sort

## Bubble Sort

the bubble sort  
algorithm  
C++ code for bubble  
sort

Insertion Sort  
the insertion sort  
algorithm

C++ code for  
insertion sort

```
5 random items : (24,b)(96,k)(12,t)(99,g)(26,w)
```

```
after 2 <-> 0 : (12,b)(96,k)(24,t)(99,g)(26,w)
```

```
after 2 <-> 1 : (12,b)(24,k)(96,t)(99,g)(26,w)
```

```
after 4 <-> 2 : (12,b)(24,k)(26,t)(99,g)(96,w)
```

```
after 4 <-> 3 : (12,b)(24,k)(26,t)(96,g)(99,w)
```

```
the sorted vector : (12,b)(24,k)(26,t)(96,g)(99,w)
```

1 Nov 2010

## running an example

Using C++  
Sorting

sorting a vector of  
pairs  
setting up C++ code

Selection Sort  
the selection sort  
algorithm

C++ code for  
selection sort

## Bubble Sort

the bubble sort  
algorithm  
C++ code for bubble  
sort

Insertion Sort  
the insertion sort  
algorithm

C++ code for  
insertion sort

5 random items : (24,b)(96,k)(12,t)(99,g)(26,w)

after 2 <-> 0 : (12,b)(96,k)(24,t)(99,g)(26,w)

after 2 <-> 1 : (12,b)(24,k)(96,t)(99,g)(26,w)

after 4 <-> 2 : (12,b)(24,k)(26,t)(99,g)(96,w)

after 4 <-> 3 : (12,b)(24,k)(26,t)(96,g)(99,w)

the sorted vector : (12,b)(24,k)(26,t)(96,g)(99,w)

1 Nov 2010

## running an example

Using C++  
Sortingsorting a vector of  
pairs

setting up C++ code

## Selection Sort

the selection sort  
algorithmC++ code for  
selection sort

## Bubble Sort

the bubble sort  
algorithmC++ code for bubble  
sort

## Insertion Sort

the insertion sort  
algorithmC++ code for  
insertion sort

```
5 random items : (24,b)(96,k)(12,t)(99,g)(26,w)
```

```
after 2 <-> 0 : (12,b)(96,k)(24,t)(99,g)(26,w)
```

```
after 2 <-> 1 : (12,b)(24,k)(96,t)(99,g)(26,w)
```

```
after 4 <-> 2 : (12,b)(24,k)(26,t)(99,g)(96,w)
```

```
after 4 <-> 3 : (12,b)(24,k)(26,t)(96,g)(99,w)
```

```
the sorted vector : (12,b)(24,k)(26,t)(96,g)(99,w)
```

1 Nov 2010

## running an example

Using C++  
Sortingsorting a vector of  
pairs

setting up C++ code

## Selection Sort

the selection sort  
algorithmC++ code for  
selection sort

## Bubble Sort

the bubble sort  
algorithmC++ code for bubble  
sort

## Insertion Sort

the insertion sort  
algorithmC++ code for  
insertion sort

```
5 random items : (24,b)(96,k)(12,t)(99,g)(26,w)
```

```
after 2 <-> 0 : (12,b)(96,k)(24,t)(99,g)(26,w)
```

```
after 2 <-> 1 : (12,b)(24,k)(96,t)(99,g)(26,w)
```

```
after 4 <-> 2 : (12,b)(24,k)(26,t)(99,g)(96,w)
```

```
after 4 <-> 3 : (12,b)(24,k)(26,t)(96,g)(99,w)
```

```
the sorted vector : (12,b)(24,k)(26,t)(96,g)(99,w)
```

1 Nov 2010

## running an example

Using C++  
Sortingsorting a vector of  
pairs

setting up C++ code

## Selection Sort

the selection sort  
algorithmC++ code for  
selection sort

## Bubble Sort

the bubble sort  
algorithmC++ code for bubble  
sort

## Insertion Sort

the insertion sort  
algorithmC++ code for  
insertion sort

```
5 random items : (24,b)(96,k)(12,t)(99,g)(26,w)
```

```
after 2 <-> 0 : (12,b)(96,k)(24,t)(99,g)(26,w)
```

```
after 2 <-> 1 : (12,b)(24,k)(96,t)(99,g)(26,w)
```

```
after 4 <-> 2 : (12,b)(24,k)(26,t)(99,g)(96,w)
```

```
after 4 <-> 3 : (12,b)(24,k)(26,t)(96,g)(99,w)
```

```
the sorted vector : (12,b)(24,k)(26,t)(96,g)(99,w)
```

# Quadratic Sorting Algorithms

## Using C++ Sorting

sorting a vector of  
pairs  
setting up C++ code

## Selection Sort

the selection sort  
algorithm

C++ code for  
selection sort

## Bubble Sort

the bubble sort  
algorithm  
C++ code for bubble  
sort

## Insertion Sort

the insertion sort  
algorithm  
C++ code for  
insertion sort

- 1 Using C++ Sorting  
sorting a vector of pairs  
setting up C++ code
- 2 Selection Sort  
the selection sort algorithm  
C++ code for selection sort
- 3 Bubble Sort  
the bubble sort algorithm  
C++ code for bubble sort
- 4 Insertion Sort  
the insertion sort algorithm  
C++ code for insertion sort

## code for selection sort

```
void sort ( vector< pair<int, char> >& v )
{
    for(int i=0; i<v.size()-1; i++)
    {
        int m = v[i].first;
        int k = i;
        for(int j=i+1; j<v.size(); j++)
            if(v[j].first < m)
            {
                m = v[j].first; k = j;
            }
        if(k != i)
        {
            v[k].first = v[i].first;
            v[i].first = m;
        }
    }
}
```

Using C++  
Sorting

sorting a vector of  
pairs

setting up C++ code

## Selection Sort

the selection sort  
algorithm

C++ code for  
selection sort

## Bubble Sort

the bubble sort  
algorithm

C++ code for bubble  
sort

## Insertion Sort

the insertion sort  
algorithm

C++ code for  
insertion sort

## code for selection sort

```
void sort ( vector< pair<int, char> >& v )
{
    for(int i=0; i<v.size()-1; i++)
    {
        int m = v[i].first;
        int k = i;
        for(int j=i+1; j<v.size(); j++)
            if(v[j].first < m)
            {
                m = v[j].first; k = j;
            }
        if(k != i)
        {
            v[k].first = v[i].first;
            v[i].first = m;
        }
    }
}
```

Using C++  
Sortingsorting a vector of  
pairs

setting up C++ code

## Selection Sort

the selection sort  
algorithmC++ code for  
selection sort

## Bubble Sort

the bubble sort  
algorithmC++ code for bubble  
sort

## Insertion Sort

the insertion sort  
algorithmC++ code for  
insertion sort

## code for selection sort

```
void sort ( vector< pair<int, char> >& v )
{
    for(int i=0; i<v.size()-1; i++)
    {
        int m = v[i].first;
        int k = i;
        for(int j=i+1; j<v.size(); j++)
            if(v[j].first < m)
            {
                m = v[j].first; k = j;
            }
        if(k != i)
        {
            v[k].first = v[i].first;
            v[i].first = m;
        }
    }
}
```

Using C++  
Sorting

sorting a vector of  
pairs

setting up C++ code

## Selection Sort

the selection sort  
algorithm

C++ code for  
selection sort

## Bubble Sort

the bubble sort  
algorithm

C++ code for bubble  
sort

## Insertion Sort

the insertion sort  
algorithm

C++ code for  
insertion sort

# Cost Analysis

## Using C++ Sorting

sorting a vector of  
pairs  
setting up C++ code

## Selection Sort

the selection sort  
algorithm

C++ code for  
selection sort

## Bubble Sort

the bubble sort  
algorithm

C++ code for bubble  
sort

## Insertion Sort

the insertion sort  
algorithm

C++ code for  
insertion sort

The cost of a sort depends on

- 1 the number of comparisons,
- 2 the number of assignments.

Looking at relevant parts of code:

```
for(int i=0; i<v.size()-1; i++)
    for(int j=i+1; j<v.size(); j++)
        if(v[j].first < m)
```

Let  $n = v.size()$ , then  $n - i - 1$  comparisons are performed for  $i$  from 0 to  $n - 1$ .

#comparisons:  $n - 1 + n - 2 + \dots + 2 + 1 = \frac{1}{2}n(n - 1)$

Worst case for swaps?  $v$  is given in reverse order.

Best case?  $v$  is already sorted.

# Cost Analysis

## Using C++ Sorting

sorting a vector of  
pairs  
setting up C++ code

## Selection Sort

the selection sort  
algorithm

C++ code for  
selection sort

## Bubble Sort

the bubble sort  
algorithm

C++ code for bubble  
sort

## Insertion Sort

the insertion sort  
algorithm

C++ code for  
insertion sort

The cost of a sort depends on

- 1 the number of comparisons,
- 2 the number of assignments.

Looking at relevant parts of code:

```
for(int i=0; i<v.size()-1; i++)
    for(int j=i+1; j<v.size(); j++)
        if(v[j].first < m)
```

Let  $n = v.size()$ , then  $n - i - 1$  comparisons are performed for  $i$  from 0 to  $n - 1$ .

#comparisons:  $n - 1 + n - 2 + \dots + 2 + 1 = \frac{1}{2}n(n - 1)$

Worst case for swaps?  $v$  is given in reverse order.

Best case?  $v$  is already sorted.

# Cost Analysis

## Using C++ Sorting

sorting a vector of  
pairs  
setting up C++ code

## Selection Sort

the selection sort  
algorithm

C++ code for  
selection sort

## Bubble Sort

the bubble sort  
algorithm

C++ code for bubble  
sort

## Insertion Sort

the insertion sort  
algorithm

C++ code for  
insertion sort

The cost of a sort depends on

- 1 the number of comparisons,
- 2 the number of assignments.

Looking at relevant parts of code:

```
for(int i=0; i<v.size()-1; i++)
    for(int j=i+1; j<v.size(); j++)
        if(v[j].first < m)
```

Let  $n = v.size()$ , then  $n - i - 1$  comparisons are performed for  $i$  from 0 to  $n - 1$ .

#comparisons:  $n - 1 + n - 2 + \dots + 2 + 1 = \frac{1}{2}n(n - 1)$

Worst case for swaps?  $v$  is given in reverse order.

Best case?  $v$  is already sorted.

# Cost Analysis

## Using C++ Sorting

sorting a vector of  
pairs  
setting up C++ code

## Selection Sort

the selection sort  
algorithm

C++ code for  
selection sort

## Bubble Sort

the bubble sort  
algorithm

C++ code for bubble  
sort

## Insertion Sort

the insertion sort  
algorithm

C++ code for  
insertion sort

The cost of a sort depends on

- 1 the number of comparisons,
- 2 the number of assignments.

Looking at relevant parts of code:

```
for(int i=0; i<v.size()-1; i++)
    for(int j=i+1; j<v.size(); j++)
        if(v[j].first < m)
```

Let  $n = v.size()$ , then  $n - i - 1$  comparisons are performed for  $i$  from 0 to  $n - 1$ .

#comparisons:  $n - 1 + n - 2 + \dots + 2 + 1 = \frac{1}{2}n(n - 1)$

Worst case for swaps?  $v$  is given in reverse order.

Best case?  $v$  is already sorted.

# Quadratic Sorting Algorithms

## Using C++ Sorting

sorting a vector of  
pairs  
setting up C++ code

## Selection Sort

the selection sort  
algorithm  
C++ code for  
selection sort

## Bubble Sort

the bubble sort  
algorithm  
C++ code for bubble  
sort

## Insertion Sort

the insertion sort  
algorithm  
C++ code for  
insertion sort

- 1 Using C++ Sorting  
sorting a vector of pairs  
setting up C++ code
- 2 Selection Sort  
the selection sort algorithm  
C++ code for selection sort
- 3 Bubble Sort  
the bubble sort algorithm  
C++ code for bubble sort
- 4 Insertion Sort  
the insertion sort algorithm  
C++ code for insertion sort

# Bubble Sort

## Using C++ Sorting

sorting a vector of  
pairs  
setting up C++ code

## Selection Sort

the selection sort  
algorithm  
C++ code for  
selection sort

## Bubble Sort

the bubble sort  
algorithm  
C++ code for bubble  
sort

## Insertion Sort

the insertion sort  
algorithm  
C++ code for  
insertion sort

## How verify that a vector is already sorted?

We run through the sequence:

as long as two consecutive elements are out of order: swap.

Largest element get swapped to the end.

For  $i$  from 0 to  $n - 1$  do

- 1 assume sequence is sorted;
- 2 for  $j$  from 0 to  $n - i - 1$  do
  - if  $v_j > v_{j+1}$  then swap  $v_j \leftrightarrow v_{j+1}$   
and sequence was not sorted;
- 3 if no swap occurred, then leave the loop.

Loop invariant: last  $i$  elements are largest.

# Bubble Sort

## Using C++ Sorting

sorting a vector of  
pairs  
setting up C++ code

## Selection Sort

the selection sort  
algorithm  
C++ code for  
selection sort

## Bubble Sort

the bubble sort  
algorithm  
C++ code for bubble  
sort

## Insertion Sort

the insertion sort  
algorithm  
C++ code for  
insertion sort

How verify that a vector is already sorted?

We run through the sequence:  
as long as two consecutive elements are out of order: swap.

Largest element get swapped to the end.

For  $i$  from 0 to  $n - 1$  do

- 1 assume sequence is sorted;
- 2 for  $j$  from 0 to  $n - i - 1$  do
  - if  $v_j > v_{j+1}$  then swap  $v_j \leftrightarrow v_{j+1}$   
and sequence was not sorted;
- 3 if no swap occurred, then leave the loop.

Loop invariant: last  $i$  elements are largest.

# Bubble Sort

## Using C++ Sorting

sorting a vector of  
pairs  
setting up C++ code

## Selection Sort

the selection sort  
algorithm  
C++ code for  
selection sort

## Bubble Sort

the bubble sort  
algorithm  
C++ code for bubble  
sort

## Insertion Sort

the insertion sort  
algorithm  
C++ code for  
insertion sort

How verify that a vector is already sorted?

We run through the sequence:

as long as two consecutive elements are out of order: swap.

Largest element get swapped to the end.

For  $i$  from 0 to  $n - 1$  do

- 1 assume sequence is sorted;
- 2 for  $j$  from 0 to  $n - i - 1$  do
  - if  $v_j > v_{j+1}$  then swap  $v_j \leftrightarrow v_{j+1}$   
and sequence was not sorted;
- 3 if no swap occurred, then leave the loop.

Loop invariant: last  $i$  elements are largest.

# Bubble Sort

## Using C++ Sorting

sorting a vector of  
pairs  
setting up C++ code

## Selection Sort

the selection sort  
algorithm  
C++ code for  
selection sort

## Bubble Sort

the bubble sort  
algorithm  
C++ code for bubble  
sort

## Insertion Sort

the insertion sort  
algorithm  
C++ code for  
insertion sort

How verify that a vector is already sorted?

We run through the sequence:

as long as two consecutive elements are out of order: swap.

Largest element get swapped to the end.

For  $i$  from 0 to  $n - 1$  do

- 1 assume sequence is sorted;
- 2 for  $j$  from 0 to  $n - i - 1$  do
  - if  $v_j > v_{j+1}$  then swap  $v_j \leftrightarrow v_{j+1}$   
and sequence was not sorted;
- 3 if no swap occurred, then leave the loop.

Loop invariant: last  $i$  elements are largest.

## running an example

Using C++  
Sorting

sorting a vector of  
pairs  
setting up C++ code

## Selection Sort

the selection sort  
algorithm  
C++ code for  
selection sort

## Bubble Sort

the bubble sort  
algorithm  
C++ code for bubble  
sort

## Insertion Sort

the insertion sort  
algorithm  
C++ code for  
insertion sort

5 random items : (62,r)(52,y)(30,k)(70,l)(31,e)

after 0 <-> 1 : (52,y)(62,r)(30,k)(70,l)(31,e)

after 1 <-> 2 : (52,y)(30,k)(62,r)(70,l)(31,e)

after 3 <-> 4 : (52,y)(30,k)(62,r)(31,e)(70,l)

after 0 <-> 1 : (30,k)(52,y)(62,r)(31,e)(70,l)

after 2 <-> 3 : (30,k)(52,y)(31,e)(62,r)(70,l)

after 1 <-> 2 : (30,k)(31,e)(52,y)(62,r)(70,l)

the sorted vector : (30,k)(31,e)(52,y)(62,r)(70,l)

1 Nov 2010

## running an example

Using C++  
Sorting

sorting a vector of  
pairs  
setting up C++ code

## Selection Sort

the selection sort  
algorithm  
C++ code for  
selection sort

## Bubble Sort

the bubble sort  
algorithm  
C++ code for bubble  
sort

## Insertion Sort

the insertion sort  
algorithm  
C++ code for  
insertion sort

5 random items : (62,r)(52,y)(30,k)(70,l)(31,e)

after 0 <-> 1 : (52,y)(62,r)(30,k)(70,l)(31,e)

after 1 <-> 2 : (52,y)(30,k)(62,r)(70,l)(31,e)

after 3 <-> 4 : (52,y)(30,k)(62,r)(31,e)(70,l)

after 0 <-> 1 : (30,k)(52,y)(62,r)(31,e)(70,l)

after 2 <-> 3 : (30,k)(52,y)(31,e)(62,r)(70,l)

after 1 <-> 2 : (30,k)(31,e)(52,y)(62,r)(70,l)

the sorted vector : (30,k)(31,e)(52,y)(62,r)(70,l)

1 Nov 2010

## running an example

Using C++  
Sorting

sorting a vector of  
pairs  
setting up C++ code

## Selection Sort

the selection sort  
algorithm  
C++ code for  
selection sort

## Bubble Sort

the bubble sort  
algorithm  
C++ code for bubble  
sort

## Insertion Sort

the insertion sort  
algorithm  
C++ code for  
insertion sort

5 random items : (62,r)(52,y)(30,k)(70,l)(31,e)

after 0 <-> 1 : (52,y)(62,r)(30,k)(70,l)(31,e)

after 1 <-> 2 : (52,y)(30,k)(62,r)(70,l)(31,e)

after 3 <-> 4 : (52,y)(30,k)(62,r)(31,e)(70,l)

after 0 <-> 1 : (30,k)(52,y)(62,r)(31,e)(70,l)

after 2 <-> 3 : (30,k)(52,y)(31,e)(62,r)(70,l)

after 1 <-> 2 : (30,k)(31,e)(52,y)(62,r)(70,l)

the sorted vector : (30,k)(31,e)(52,y)(62,r)(70,l)

1 Nov 2010

## running an example

Using C++  
Sorting

sorting a vector of  
pairs  
setting up C++ code

## Selection Sort

the selection sort  
algorithm  
C++ code for  
selection sort

## Bubble Sort

the bubble sort  
algorithm  
C++ code for bubble  
sort

## Insertion Sort

the insertion sort  
algorithm  
C++ code for  
insertion sort

5 random items : (62,r)(52,y)(30,k)(70,l)(31,e)

after 0 <-> 1 : (52,y)(62,r)(30,k)(70,l)(31,e)

after 1 <-> 2 : (52,y)(30,k)(62,r)(70,l)(31,e)

after 3 <-> 4 : (52,y)(30,k)(62,r)(31,e)(70,l)

after 0 <-> 1 : (30,k)(52,y)(62,r)(31,e)(70,l)

after 2 <-> 3 : (30,k)(52,y)(31,e)(62,r)(70,l)

after 1 <-> 2 : (30,k)(31,e)(52,y)(62,r)(70,l)

the sorted vector : (30,k)(31,e)(52,y)(62,r)(70,l)

1 Nov 2010

## running an example

Using C++  
Sorting

sorting a vector of  
pairs  
setting up C++ code

## Selection Sort

the selection sort  
algorithm  
C++ code for  
selection sort

## Bubble Sort

the bubble sort  
algorithm  
C++ code for bubble  
sort

## Insertion Sort

the insertion sort  
algorithm  
C++ code for  
insertion sort

5 random items : (62,r)(52,y)(30,k)(70,l)(31,e)

after 0 <-> 1 : (52,y)(62,r)(30,k)(70,l)(31,e)

after 1 <-> 2 : (52,y)(30,k)(62,r)(70,l)(31,e)

after 3 <-> 4 : (52,y)(30,k)(62,r)(31,e)(70,l)

after 0 <-> 1 : (30,k)(52,y)(62,r)(31,e)(70,l)

after 2 <-> 3 : (30,k)(52,y)(31,e)(62,r)(70,l)

after 1 <-> 2 : (30,k)(31,e)(52,y)(62,r)(70,l)

the sorted vector : (30,k)(31,e)(52,y)(62,r)(70,l)

1 Nov 2010

## running an example

Using C++  
Sorting

sorting a vector of  
pairs  
setting up C++ code

## Selection Sort

the selection sort  
algorithm  
C++ code for  
selection sort

## Bubble Sort

the bubble sort  
algorithm  
C++ code for bubble  
sort

## Insertion Sort

the insertion sort  
algorithm  
C++ code for  
insertion sort

5 random items : (62,r)(52,y)(30,k)(70,l)(31,e)

after 0 <-> 1 : (52,y)(62,r)(30,k)(70,l)(31,e)

after 1 <-> 2 : (52,y)(30,k)(62,r)(70,l)(31,e)

after 3 <-> 4 : (52,y)(30,k)(62,r)(31,e)(70,l)

after 0 <-> 1 : (30,k)(52,y)(62,r)(31,e)(70,l)

after 2 <-> 3 : (30,k)(52,y)(31,e)(62,r)(70,l)

after 1 <-> 2 : (30,k)(31,e)(52,y)(62,r)(70,l)

the sorted vector : (30,k)(31,e)(52,y)(62,r)(70,l)

## running an example

Using C++  
Sorting

sorting a vector of  
pairs  
setting up C++ code

## Selection Sort

the selection sort  
algorithm  
C++ code for  
selection sort

## Bubble Sort

the bubble sort  
algorithm  
C++ code for bubble  
sort

## Insertion Sort

the insertion sort  
algorithm  
C++ code for  
insertion sort

5 random items : (62,r)(52,y)(30,k)(70,l)(31,e)

after 0 <-> 1 : (52,y)(62,r)(30,k)(70,l)(31,e)

after 1 <-> 2 : (52,y)(30,k)(62,r)(70,l)(31,e)

after 3 <-> 4 : (52,y)(30,k)(62,r)(31,e)(70,l)

after 0 <-> 1 : (30,k)(52,y)(62,r)(31,e)(70,l)

after 2 <-> 3 : (30,k)(52,y)(31,e)(62,r)(70,l)

after 1 <-> 2 : (30,k)(31,e)(52,y)(62,r)(70,l)

the sorted vector : (30,k)(31,e)(52,y)(62,r)(70,l)

## running an example

Using C++  
Sorting

sorting a vector of  
pairs  
setting up C++ code

## Selection Sort

the selection sort  
algorithm  
C++ code for  
selection sort

## Bubble Sort

the bubble sort  
algorithm  
C++ code for bubble  
sort

## Insertion Sort

the insertion sort  
algorithm  
C++ code for  
insertion sort

5 random items : (62,r)(52,y)(30,k)(70,l)(31,e)

after 0 <-> 1 : (52,y)(62,r)(30,k)(70,l)(31,e)

after 1 <-> 2 : (52,y)(30,k)(62,r)(70,l)(31,e)

after 3 <-> 4 : (52,y)(30,k)(62,r)(31,e)(70,l)

after 0 <-> 1 : (30,k)(52,y)(62,r)(31,e)(70,l)

after 2 <-> 3 : (30,k)(52,y)(31,e)(62,r)(70,l)

after 1 <-> 2 : (30,k)(31,e)(52,y)(62,r)(70,l)

the sorted vector : (30,k)(31,e)(52,y)(62,r)(70,l)

# Quadratic Sorting Algorithms

## Using C++ Sorting

sorting a vector of  
pairs  
setting up C++ code

## Selection Sort

the selection sort  
algorithm  
C++ code for  
selection sort

## Bubble Sort

the bubble sort  
algorithm  
C++ code for bubble  
sort

## Insertion Sort

the insertion sort  
algorithm  
C++ code for  
insertion sort

- 1 Using C++ Sorting  
sorting a vector of pairs  
setting up C++ code
- 2 Selection Sort  
the selection sort algorithm  
C++ code for selection sort
- 3 Bubble Sort  
the bubble sort algorithm  
C++ code for bubble sort
- 4 Insertion Sort  
the insertion sort algorithm  
C++ code for insertion sort

## code for bubble sort

```
void sort ( vector< pair<int,char> >& v )
{
    for(int i=0; i<v.size()-1; i++)
    {
        bool sorted = true;

        for(int j=0; j<v.size()-i-1; j++)
            if(v[j].first > v[j+1].first)
            {
                pair<int,char> tmp = v[j];
                v[j] = v[j+1];
                v[j+1] = tmp;

                sorted = false;
            }

        if(sorted) break;
    }
}
```

Using C++  
Sorting

sorting a vector of  
pairs

setting up C++ code

## Selection Sort

the selection sort  
algorithm

C++ code for  
selection sort

## Bubble Sort

the bubble sort  
algorithm

C++ code for bubble  
sort

## Insertion Sort

the insertion sort  
algorithm

C++ code for  
insertion sort

## code for bubble sort

```
void sort ( vector< pair<int,char> >& v )
{
    for(int i=0; i<v.size()-1; i++)
    {
        bool sorted = true;

        for(int j=0; j<v.size()-i-1; j++)
            if(v[j].first > v[j+1].first)
            {
                pair<int,char> tmp = v[j];
                v[j] = v[j+1];
                v[j+1] = tmp;

                sorted = false;
            }

        if(sorted) break;
    }
}
```

Using C++  
Sorting

sorting a vector of  
pairs

setting up C++ code

## Selection Sort

the selection sort  
algorithm

C++ code for  
selection sort

## Bubble Sort

the bubble sort  
algorithm

C++ code for bubble  
sort

## Insertion Sort

the insertion sort  
algorithm

C++ code for  
insertion sort

## code for bubble sort

```
void sort ( vector< pair<int,char> >& v )
{
    for(int i=0; i<v.size()-1; i++)
    {
        bool sorted = true;

        for(int j=0; j<v.size()-i-1; j++)
            if(v[j].first > v[j+1].first)
            {
                pair<int,char> tmp = v[j];
                v[j] = v[j+1];
                v[j+1] = tmp;

                sorted = false;
            }

        if(sorted) break;
    }
}
```

Using C++  
Sorting

sorting a vector of  
pairs  
setting up C++ code

## Selection Sort

the selection sort  
algorithm  
C++ code for  
selection sort

## Bubble Sort

the bubble sort  
algorithm  
C++ code for bubble  
sort

## Insertion Sort

the insertion sort  
algorithm  
C++ code for  
insertion sort

# Cost Analysis

## Using C++ Sorting

sorting a vector of  
pairs

setting up C++ code

## Selection Sort

the selection sort  
algorithm

C++ code for  
selection sort

## Bubble Sort

the bubble sort  
algorithm

C++ code for bubble  
sort

## Insertion Sort

the insertion sort  
algorithm

C++ code for  
insertion sort

Again we count comparisons and assignments.

Look at relevant parts of the code:

```
for(int i=0; i<v.size()-1; i++)
    for(int j=0; j<v.size()-i-1; j++)
        if(v[j].first > v[j+1].first)
```

Let  $n = v.size()$ , then  $n - i - 1$  comparisons are performed for  $i$  from 0 to  $n - 1$ .

#comparisons:  $n-1 + n-2 + \dots + 2 + 1 = \frac{1}{2}n(n-1)$

Bubble sort seems just as good (or bad) as selection sort, but it swaps adjacent elements and works on linked lists.

Best case? Given vector is already sorted.

Worst case for swaps?  $v$  is in reverse order.

# Cost Analysis

## Using C++ Sorting

sorting a vector of  
pairs

setting up C++ code

## Selection Sort

the selection sort  
algorithm

C++ code for  
selection sort

## Bubble Sort

the bubble sort  
algorithm

C++ code for bubble  
sort

## Insertion Sort

the insertion sort  
algorithm

C++ code for  
insertion sort

Again we count comparisons and assignments.

Look at relevant parts of the code:

```
for(int i=0; i<v.size()-1; i++)
    for(int j=0; j<v.size()-i-1; j++)
        if(v[j].first > v[j+1].first)
```

Let  $n = v.size()$ , then  $n - i - 1$  comparisons are performed for  $i$  from 0 to  $n - 1$ .

#comparisons:  $n - 1 + n - 2 + \dots + 2 + 1 = \frac{1}{2}n(n - 1)$

Bubble sort seems just as good (or bad) as selection sort, but it swaps adjacent elements and works on linked lists.

Best case? Given vector is already sorted.

Worst case for swaps?  $v$  is in reverse order.

# Cost Analysis

## Using C++ Sorting

sorting a vector of  
pairs  
setting up C++ code

## Selection Sort

the selection sort  
algorithm  
C++ code for  
selection sort

## Bubble Sort

the bubble sort  
algorithm  
C++ code for bubble  
sort

## Insertion Sort

the insertion sort  
algorithm  
C++ code for  
insertion sort

Again we count comparisons and assignments.

Look at relevant parts of the code:

```
for(int i=0; i<v.size()-1; i++)
    for(int j=0; j<v.size()-i-1; j++)
        if(v[j].first > v[j+1].first)
```

Let  $n = v.size()$ , then  $n - i - 1$  comparisons are performed for  $i$  from 0 to  $n - 1$ .

#comparisons:  $n - 1 + n - 2 + \dots + 2 + 1 = \frac{1}{2}n(n - 1)$

Bubble sort seems just as good (or bad) as selection sort, but it swaps adjacent elements and works on linked lists.

Best case? Given vector is already sorted.

Worst case for swaps?  $v$  is in reverse order.

# Quadratic Sorting Algorithms

## Using C++ Sorting

sorting a vector of  
pairs  
setting up C++ code

## Selection Sort

the selection sort  
algorithm  
C++ code for  
selection sort

## Bubble Sort

the bubble sort  
algorithm  
C++ code for bubble  
sort

## Insertion Sort

the insertion sort  
algorithm  
C++ code for  
insertion sort

- 1 Using C++ Sorting  
sorting a vector of pairs  
setting up C++ code
- 2 Selection Sort  
the selection sort algorithm  
C++ code for selection sort
- 3 Bubble Sort  
the bubble sort algorithm  
C++ code for bubble sort
- 4 Insertion Sort  
the insertion sort algorithm  
C++ code for insertion sort

Using C++  
Sorting

sorting a vector of  
pairs  
setting up C++ code

## Selection Sort

the selection sort  
algorithm  
C++ code for  
selection sort

## Bubble Sort

the bubble sort  
algorithm  
C++ code for bubble  
sort

## Insertion Sort

the insertion sort  
algorithm  
C++ code for  
insertion sort

# Insertion Sort

Imagine sorting playing cards:

- 1 hold in left hand cards already picked up,
- 2 insert newly picked card in left hand.

Denote by index  $i$  the start of the sequence of  $n$  elements still to be inserted.

For  $i$  from 0 to  $n - 1$  do

- 1 find  $j < i$ :  $v_j \leq v_i$  (if  $j = i - 1$ , then sorted);
- 2 if  $j \neq i - 1$ : shift  $v_{j+1} \dots v_{i-1}$  one spot to the right and insert  $v_i$  at the  $j$ -th spot.

Loop invariant: first  $i$  elements are sorted.

Using C++  
Sorting

sorting a vector of  
pairs  
setting up C++ code

## Selection Sort

the selection sort  
algorithm  
C++ code for  
selection sort

## Bubble Sort

the bubble sort  
algorithm  
C++ code for bubble  
sort

## Insertion Sort

the insertion sort  
algorithm  
C++ code for  
insertion sort

# Insertion Sort

Imagine sorting playing cards:

- 1 hold in left hand cards already picked up,
- 2 insert newly picked card in left hand.

Denote by index  $i$  the start of the sequence of  $n$  elements still to be inserted.

For  $i$  from 0 to  $n - 1$  do

- 1 find  $j < i$ :  $v_j \leq v_i$  (if  $j = i - 1$ , then sorted);
- 2 if  $j \neq i - 1$ : shift  $v_{j+1} \dots v_{i-1}$  one spot to the right and insert  $v_i$  at the  $j$ -th spot.

Loop invariant: first  $i$  elements are sorted.

# Insertion Sort

## Using C++ Sorting

sorting a vector of  
pairs  
setting up C++ code

## Selection Sort

the selection sort  
algorithm  
C++ code for  
selection sort

## Bubble Sort

the bubble sort  
algorithm  
C++ code for bubble  
sort

## Insertion Sort

the insertion sort  
algorithm  
C++ code for  
insertion sort

Imagine sorting playing cards:

- 1 hold in left hand cards already picked up,
- 2 insert newly picked card in left hand.

Denote by index  $i$  the start of the sequence of  $n$  elements still to be inserted.

For  $i$  from 0 to  $n - 1$  do

- 1 find  $j < i$ :  $v_j \leq v_i$  (if  $j = i - 1$ , then sorted);
- 2 if  $j \neq i - 1$ : shift  $v_{j+1} \dots v_{i-1}$  one spot to the right and insert  $v_i$  at the  $j$ -th spot.

Loop invariant: first  $i$  elements are sorted.

1 Nov 2010

## running an example

Using C++  
Sortingsorting a vector of  
pairs

setting up C++ code

## Selection Sort

the selection sort  
algorithmC++ code for  
selection sort

## Bubble Sort

the bubble sort  
algorithmC++ code for bubble  
sort

## Insertion Sort

the insertion sort  
algorithmC++ code for  
insertion sort

```
5 random items : (87,i)(48,v)(36,h)(73,a)(63,h)
```

```
after 1 <-> 0 : (48,v)(87,i)(36,h)(73,a)(63,h)
```

```
after 2 <-> 0 : (36,h)(48,v)(87,i)(73,a)(63,h)
```

```
after 3 <-> 2 : (36,h)(48,v)(73,a)(87,i)(63,h)
```

```
after 4 <-> 2 : (36,h)(48,v)(63,h)(73,a)(87,i)
```

```
the sorted vector : (36,h)(48,v)(63,h)(73,a)(87,i)
```

1 Nov 2010

## running an example

Using C++  
Sortingsorting a vector of  
pairs

setting up C++ code

## Selection Sort

the selection sort  
algorithmC++ code for  
selection sort

## Bubble Sort

the bubble sort  
algorithmC++ code for bubble  
sort

## Insertion Sort

the insertion sort  
algorithmC++ code for  
insertion sort

```
5 random items : (87,i)(48,v)(36,h)(73,a)(63,h)
```

```
after 1 <-> 0 : (48,v)(87,i)(36,h)(73,a)(63,h)
```

```
after 2 <-> 0 : (36,h)(48,v)(87,i)(73,a)(63,h)
```

```
after 3 <-> 2 : (36,h)(48,v)(73,a)(87,i)(63,h)
```

```
after 4 <-> 2 : (36,h)(48,v)(63,h)(73,a)(87,i)
```

```
the sorted vector : (36,h)(48,v)(63,h)(73,a)(87,i)
```

1 Nov 2010

## running an example

Using C++  
Sortingsorting a vector of  
pairs

setting up C++ code

## Selection Sort

the selection sort  
algorithmC++ code for  
selection sort

## Bubble Sort

the bubble sort  
algorithmC++ code for bubble  
sort

## Insertion Sort

the insertion sort  
algorithmC++ code for  
insertion sort

```
5 random items : (87,i)(48,v)(36,h)(73,a)(63,h)
```

```
after 1 <-> 0 : (48,v)(87,i)(36,h)(73,a)(63,h)
```

```
after 2 <-> 0 : (36,h)(48,v)(87,i)(73,a)(63,h)
```

```
after 3 <-> 2 : (36,h)(48,v)(73,a)(87,i)(63,h)
```

```
after 4 <-> 2 : (36,h)(48,v)(63,h)(73,a)(87,i)
```

```
the sorted vector : (36,h)(48,v)(63,h)(73,a)(87,i)
```

1 Nov 2010

## running an example

Using C++  
Sortingsorting a vector of  
pairs

setting up C++ code

## Selection Sort

the selection sort  
algorithmC++ code for  
selection sort

## Bubble Sort

the bubble sort  
algorithmC++ code for bubble  
sort

## Insertion Sort

the insertion sort  
algorithmC++ code for  
insertion sort

```
5 random items : (87,i)(48,v)(36,h)(73,a)(63,h)
```

```
after 1 <-> 0 : (48,v)(87,i)(36,h)(73,a)(63,h)
```

```
after 2 <-> 0 : (36,h)(48,v)(87,i)(73,a)(63,h)
```

```
after 3 <-> 2 : (36,h)(48,v)(73,a)(87,i)(63,h)
```

```
after 4 <-> 2 : (36,h)(48,v)(63,h)(73,a)(87,i)
```

```
the sorted vector : (36,h)(48,v)(63,h)(73,a)(87,i)
```

## running an example

Using C++  
Sorting

sorting a vector of  
pairs

setting up C++ code

## Selection Sort

the selection sort  
algorithm

C++ code for  
selection sort

## Bubble Sort

the bubble sort  
algorithm

C++ code for bubble  
sort

## Insertion Sort

the insertion sort  
algorithm

C++ code for  
insertion sort

```
5 random items : (87,i)(48,v)(36,h)(73,a)(63,h)
```

```
after 1 <-> 0 : (48,v)(87,i)(36,h)(73,a)(63,h)
```

```
after 2 <-> 0 : (36,h)(48,v)(87,i)(73,a)(63,h)
```

```
after 3 <-> 2 : (36,h)(48,v)(73,a)(87,i)(63,h)
```

```
after 4 <-> 2 : (36,h)(48,v)(63,h)(73,a)(87,i)
```

```
the sorted vector : (36,h)(48,v)(63,h)(73,a)(87,i)
```

## running an example

Using C++  
Sorting

sorting a vector of  
pairs  
setting up C++ code

## Selection Sort

the selection sort  
algorithm

C++ code for  
selection sort

## Bubble Sort

the bubble sort  
algorithm

C++ code for bubble  
sort

## Insertion Sort

the insertion sort  
algorithm

C++ code for  
insertion sort

```
5 random items : (87,i)(48,v)(36,h)(73,a)(63,h)
```

```
after 1 <-> 0 : (48,v)(87,i)(36,h)(73,a)(63,h)
```

```
after 2 <-> 0 : (36,h)(48,v)(87,i)(73,a)(63,h)
```

```
after 3 <-> 2 : (36,h)(48,v)(73,a)(87,i)(63,h)
```

```
after 4 <-> 2 : (36,h)(48,v)(63,h)(73,a)(87,i)
```

```
the sorted vector : (36,h)(48,v)(63,h)(73,a)(87,i)
```

# Quadratic Sorting Algorithms

## Using C++ Sorting

sorting a vector of  
pairs  
setting up C++ code

## Selection Sort

the selection sort  
algorithm  
C++ code for  
selection sort

## Bubble Sort

the bubble sort  
algorithm  
C++ code for bubble  
sort

## Insertion Sort

the insertion sort  
algorithm  
C++ code for  
insertion sort

- 1 Using C++ Sorting  
sorting a vector of pairs  
setting up C++ code
- 2 Selection Sort  
the selection sort algorithm  
C++ code for selection sort
- 3 Bubble Sort  
the bubble sort algorithm  
C++ code for bubble sort
- 4 Insertion Sort  
the insertion sort algorithm  
C++ code for insertion sort

## code for insertion sort

Using C++  
Sorting

sorting a vector of  
pairs  
setting up C++ code

## Selection Sort

the selection sort  
algorithm  
C++ code for  
selection sort

## Bubble Sort

the bubble sort  
algorithm  
C++ code for bubble  
sort

## Insertion Sort

the insertion sort  
algorithm  
C++ code for  
insertion sort

```
void sort ( vector< pair<int,char> >& v )
{
    for(int i=1; i<v.size(); i++)
    {
        int j=i-1;
        while((j >= 0)
            && (v[i].first < v[j].first)) j--;

        if(++j < i)
        {
            pair<int,char> tmp = v[i];
            for(int k=i-1; k>=j; k--) v[k+1] = v[k];
            v[j] = tmp;
        }
    }
}
```

## code for insertion sort

Using C++  
Sorting

sorting a vector of  
pairs  
setting up C++ code

## Selection Sort

the selection sort  
algorithm  
C++ code for  
selection sort

## Bubble Sort

the bubble sort  
algorithm  
C++ code for bubble  
sort

## Insertion Sort

the insertion sort  
algorithm  
C++ code for  
insertion sort

```
void sort ( vector< pair<int,char> >& v )
{
    for(int i=1; i<v.size(); i++)
    {
        int j=i-1;
        while((j >= 0)
            && (v[i].first < v[j].first)) j--;

        if(++j < i)
        {
            pair<int,char> tmp = v[i];
            for(int k=i-1; k>=j; k--) v[k+1] = v[k];
            v[j] = tmp;
        }
    }
}
```

# Cost Analysis

## Using C++ Sorting

sorting a vector of  
pairs  
setting up C++ code

## Selection Sort

the selection sort  
algorithm  
C++ code for  
selection sort

## Bubble Sort

the bubble sort  
algorithm  
C++ code for bubble  
sort

## Insertion Sort

the insertion sort  
algorithm  
C++ code for  
insertion sort

Once more we count comparisons and assignments.

We look at relevant part of the code:

```
for(int i=1; i<v.size(); i++)
{
    int j=i-1;
    while((j >= 0)
        && (v[i].first < v[j].first)) j--;
```

Denote  $n = v.size()$ , we insert  $n - 1$  elements  
and in the worst case we make  
 $n - 1 + n - 2 + \dots + 2 + 1 = \frac{1}{2}n(n - 1)$  comparisons.

In the best case, when sorted:  $n - 1$  comparisons.  
Cost of shifting elements is smaller with linked list.

# Cost Analysis

## Using C++ Sorting

sorting a vector of  
pairs  
setting up C++ code

## Selection Sort

the selection sort  
algorithm  
C++ code for  
selection sort

## Bubble Sort

the bubble sort  
algorithm  
C++ code for bubble  
sort

## Insertion Sort

the insertion sort  
algorithm  
C++ code for  
insertion sort

Once more we count comparisons and assignments.

We look at relevant part of the code:

```
for(int i=1; i<v.size(); i++)
{
    int j=i-1;
    while((j >= 0)
        && (v[i].first < v[j].first)) j--;
```

Denote  $n = v.size()$ , we insert  $n - 1$  elements  
and in the worst case we make  
 $n - 1 + n - 2 + \dots + 2 + 1 = \frac{1}{2}n(n - 1)$  comparisons.

In the best case, when sorted:  $n - 1$  comparisons.  
Cost of shifting elements is smaller with linked list.

Using C++  
Sorting

sorting a vector of  
pairs  
setting up C++ code

## Selection Sort

the selection sort  
algorithm  
C++ code for  
selection sort

## Bubble Sort

the bubble sort  
algorithm  
C++ code for bubble  
sort

## Insertion Sort

the insertion sort  
algorithm  
C++ code for  
insertion sort

Once more we count comparisons and assignments.

We look at relevant part of the code:

```
for(int i=1; i<v.size(); i++)
{
    int j=i-1;
    while((j >= 0)
        && (v[i].first < v[j].first)) j--;
```

Denote  $n = v.size()$ , we insert  $n - 1$  elements  
and in the worst case we make  
 $n - 1 + n - 2 + \dots + 2 + 1 = \frac{1}{2}n(n - 1)$  comparisons.

In the best case, when sorted:  $n - 1$  comparisons.  
Cost of shifting elements is smaller with linked list.

# comparison of quadratic sorts

## Using C++ Sorting

sorting a vector of  
pairs

setting up C++ code

## Selection Sort

the selection sort  
algorithm

C++ code for  
selection sort

## Bubble Sort

the bubble sort  
algorithm

C++ code for bubble  
sort

## Insertion Sort

the insertion sort  
algorithm

C++ code for  
insertion sort

sort cases	#comparisons		#exchanges	
	best	worst	best	worst
selection	$O(n^2)$	$O(n^2)$	$O(1)$	$O(n)$
bubble	$O(n)$	$O(n^2)$	$O(1)$	$O(n^2)$
insertion	$O(n)$	$O(n^2)$	$O(n)$	$O(n^2)$

# comparison of quadratic sorts

## Using C++ Sorting

sorting a vector of  
pairs

setting up C++ code

## Selection Sort

the selection sort  
algorithm

C++ code for  
selection sort

## Bubble Sort

the bubble sort  
algorithm

C++ code for bubble  
sort

## Insertion Sort

the insertion sort  
algorithm

C++ code for  
insertion sort

sort cases	#comparisons		#exchanges	
	best	worst	best	worst
selection	$O(n^2)$	$O(n^2)$	$O(1)$	$O(n)$
bubble	$O(n)$	$O(n^2)$	$O(1)$	$O(n^2)$
insertion	$O(n)$	$O(n^2)$	$O(n)$	$O(n^2)$

# comparison of quadratic sorts

## Using C++ Sorting

sorting a vector of  
pairs

setting up C++ code

## Selection Sort

the selection sort  
algorithm

C++ code for  
selection sort

## Bubble Sort

the bubble sort  
algorithm

C++ code for bubble  
sort

## Insertion Sort

the insertion sort  
algorithm

C++ code for  
insertion sort

sort cases	#comparisons		#exchanges	
	best	worst	best	worst
selection	$O(n^2)$	$O(n^2)$	$O(1)$	$O(n)$
bubble	$O(n)$	$O(n^2)$	$O(1)$	$O(n^2)$
insertion	$O(n)$	$O(n^2)$	$O(n)$	$O(n^2)$

# Summary + Assignments

## Using C++ Sorting

sorting a vector of  
pairs  
setting up C++ code

## Selection Sort

the selection sort  
algorithm  
C++ code for  
selection sort

## Bubble Sort

the bubble sort  
algorithm  
C++ code for bubble  
sort

## Insertion Sort

the insertion sort  
algorithm  
C++ code for  
insertion sort

Started chapter 10 on sorting algorithms with quadratic cost: selection, bubble, and insertion sort.

## Assignments:

- 1 Write a recursive function for selection sort.
- 2 Give code for a bubble sort on a linked list.
- 3 Change the code for insertion sort to take on input random-access iterators to begin and end of a vector.
- 4 Modify the sort functions to count the `#comparisons` and `#swaps`. Run 10 tests on random vectors of length 100 with selection, bubble, and insertion sorts. Record the `#comparisons` and `#swaps` in a table.

**Homework collection on Monday 7 November, at noon:**  
#2 of L-24, #1, 3 of L-25, #1, 2 of L-26.