

# quicksort

## Set Up

sorting a vector of pairs

## Quicksort

the quicksort algorithm  
C++ code for quicksort

## Partition

partitioning a vector  
code for partition

## Iterators and Timers

partition with iterator  
timing C++ programs

- 1 Set Up  
sorting a vector of pairs
- 2 Quicksort  
the quicksort algorithm  
C++ code for quicksort
- 3 Partition  
partitioning a vector  
code for partition
- 4 Iterators and Timers  
partition with iterator  
timing C++ programs

MCS 360 Lecture 32  
Introduction to Data Structures  
Jan Vershelde, 5 November 2010

5 Nov 2010

Set Up

sorting a vector of pairs

Quicksort

the quicksort algorithm  
C++ code for quicksort

Partition

partitioning a vector  
code for partition

Iterators and Timers

partition with iterator  
timing C++ programs

# quicksort

- 1 Set Up  
sorting a vector of pairs
- 2 Quicksort  
the quicksort algorithm  
C++ code for quicksort
- 3 Partition  
partitioning a vector  
code for partition
- 4 Iterators and Timers  
partition with iterator  
timing C++ programs

# sorting a vector of pairs

## Set Up

sorting a vector of  
pairs

## Quicksort

the quicksort  
algorithm  
C++ code for  
quicksort

## Partition

partitioning a vector  
code for partition

## Iterators and Timers

partition with iterator  
timing C++ programs

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

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

With `char` as second data field we can check if the sort is stable.

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

We generate random vectors of 2-digit keys.

# sorting a vector of pairs

## Set Up

sorting a vector of  
pairs

## Quicksort

the quicksort  
algorithm  
C++ code for  
quicksort

## Partition

partitioning a vector  
code for partition

## Iterators and Timers

partition with iterator  
timing C++ programs

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

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

With `char` as second data field we can check if the sort is stable.

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

We generate random vectors of 2-digit keys.

5 Nov 2010

Set Up

sorting a vector of pairs

Quicksort

the quicksort algorithm

C++ code for quicksort

Partition

partitioning a vector  
code for partition

Iterators and Timers

partition with iterator  
timing C++ programs

# quicksort

- 1 Set Up  
sorting a vector of pairs
- 2 **Quicksort**  
**the quicksort algorithm**  
C++ code for quicksort
- 3 Partition  
partitioning a vector  
code for partition
- 4 Iterators and Timers  
partition with iterator  
timing C++ programs

## Set Up

sorting a vector of  
pairs

## Quicksort

the quicksort  
algorithm

C++ code for  
quicksort

## Partition

partitioning a vector  
code for partition

Iterators and  
Timers

partition with iterator  
timing C++ programs

Let  $n$  be the length of the sequence to sort.

A recursive sorting algorithm, for  $n > 1$ :

- 1 Partition the sequence in two halves:
  - 1 select a pivot,
  - 2 elements in first half  $\leq$  pivot,
  - 3 elements in second half  $>$  pivot.
- 2 Apply quicksort to the first half.
- 3 Apply quicksort to the second half.

## Set Up

sorting a vector of  
pairs

## Quicksort

the quicksort  
algorithm

C++ code for  
quicksort

## Partition

partitioning a vector  
code for partition

Iterators and  
Timers

partition with iterator  
timing C++ programs

Let  $n$  be the length of the sequence to sort.

A recursive sorting algorithm, for  $n > 1$ :

- 1 Partition the sequence in two halves:
  - 1 select a pivot,
  - 2 elements in first half  $\leq$  pivot,
  - 3 elements in second half  $>$  pivot.
- 2 Apply quicksort to the first half.
- 3 Apply quicksort to the second half.

## Set Up

sorting a vector of  
pairs

## Quicksort

the quicksort  
algorithm

C++ code for  
quicksort

## Partition

partitioning a vector  
code for partition

Iterators and  
Timers

partition with iterator  
timing C++ programs

Let  $n$  be the length of the sequence to sort.

A recursive sorting algorithm, for  $n > 1$ :

- 1 Partition the sequence in two halves:
  - 1 select a pivot,
  - 2 elements in first half  $\leq$  pivot,
  - 3 elements in second half  $>$  pivot.
- 2 Apply quicksort to the first half.
- 3 Apply quicksort to the second half.

## Set Up

sorting a vector of  
pairs

## Quicksort

the quicksort  
algorithm

C++ code for  
quicksort

## Partition

partitioning a vector  
code for partition

Iterators and  
Timers

partition with iterator  
timing C++ programs

Let  $n$  be the length of the sequence to sort.

A recursive sorting algorithm, for  $n > 1$ :

- 1 Partition the sequence in two halves:
  - 1 select a pivot,
  - 2 elements in first half  $\leq$  pivot,
  - 3 elements in second half  $>$  pivot.
- 2 Apply quicksort to the first half.
- 3 Apply quicksort to the second half.

5 Nov 2010

## running an example

## Set Up

sorting a vector of  
pairs

## Quicksort

the quicksort  
algorithmC++ code for  
quicksort

## Partition

partitioning a vector  
code for partitionIterators and  
Timerspartition with iterator  
timing C++ programs

5 random items : (74,v)(86,e)(65,n)(99,c)(43,j)

(74,v)(86,e)(65,n)(99,c)(43,j)

(65,n)(43,j)

(43,j)

(86,e)(99,c)

(99,c)

the sorted vector : (43,j)(65,n)(74,v)(86,e)(99,c)

## running an example

## Set Up

sorting a vector of  
pairs

## Quicksort

the quicksort  
algorithm

C++ code for  
quicksort

## Partition

partitioning a vector  
code for partition

Iterators and  
Timers

partition with iterator  
timing C++ programs

5 random items : (74,v)(86,e)(65,n)(99,c)(43,j)

(74,v)(86,e)(65,n)(99,c)(43,j)

(65,n)(43,j)

(43,j)

(86,e)(99,c)

(99,c)

the sorted vector : (43,j)(65,n)(74,v)(86,e)(99,c)

## running an example

## Set Up

sorting a vector of  
pairs

## Quicksort

the quicksort  
algorithm

C++ code for  
quicksort

## Partition

partitioning a vector  
code for partition

Iterators and  
Timers

partition with iterator  
timing C++ programs

5 random items : (74,v)(86,e)(65,n)(99,c)(43,j)

(74,v)(86,e)(65,n)(99,c)(43,j)

(65,n)(43,j)

(43,j)

(86,e)(99,c)

(99,c)

the sorted vector : (43,j)(65,n)(74,v)(86,e)(99,c)

## running an example

## Set Up

sorting a vector of  
pairs

## Quicksort

the quicksort  
algorithm

C++ code for  
quicksort

## Partition

partitioning a vector  
code for partition

Iterators and  
Timers

partition with iterator  
timing C++ programs

5 random items : (74,v)(86,e)(65,n)(99,c)(43,j)

(74,v)(86,e)(65,n)(99,c)(43,j)

(65,n)(43,j)

(43,j)

(86,e)(99,c)

(99,c)

the sorted vector : (43,j)(65,n)(74,v)(86,e)(99,c)

## running an example

## Set Up

sorting a vector of  
pairs

## Quicksort

the quicksort  
algorithm

C++ code for  
quicksort

## Partition

partitioning a vector  
code for partition

Iterators and  
Timers

partition with iterator  
timing C++ programs

5 random items : (74,v)(86,e)(65,n)(99,c)(43,j)

(74,v)(86,e)(65,n)(99,c)(43,j)

(65,n)(43,j)

(43,j)

(86,e)(99,c)

(99,c)

the sorted vector : (43,j)(65,n)(74,v)(86,e)(99,c)

5 Nov 2010

## running an example

## Set Up

sorting a vector of  
pairs

## Quicksort

the quicksort  
algorithmC++ code for  
quicksort

## Partition

partitioning a vector  
code for partitionIterators and  
Timerspartition with iterator  
timing C++ programs

```
5 random items : (74,v)(86,e)(65,n)(99,c)(43,j)
```

```
(74,v)(86,e)(65,n)(99,c)(43,j)
```

```
(65,n)(43,j)
```

```
(43,j)
```

```
(86,e)(99,c)
```

```
(99,c)
```

```
the sorted vector : (43,j)(65,n)(74,v)(86,e)(99,c)
```

## running an example

## Set Up

sorting a vector of  
pairs

## Quicksort

the quicksort  
algorithm

C++ code for  
quicksort

## Partition

partitioning a vector  
code for partition

Iterators and  
Timers

partition with iterator  
timing C++ programs

5 random items : (74,v)(86,e)(65,n)(99,c)(43,j)

(74,v)(86,e)(65,n)(99,c)(43,j)

(65,n)(43,j)

(43,j)

(86,e)(99,c)

(99,c)

the sorted vector : (43,j)(65,n)(74,v)(86,e)(99,c)

5 Nov 2010

Set Up

sorting a vector of pairs

Quicksort

the quicksort algorithm

C++ code for quicksort

Partition

partitioning a vector  
code for partition

Iterators and Timers

partition with iterator  
timing C++ programs

# quicksort

- 1 Set Up  
sorting a vector of pairs
- 2 Quicksort  
the quicksort algorithm  
C++ code for quicksort
- 3 Partition  
partitioning a vector  
code for partition
- 4 Iterators and Timers  
partition with iterator  
timing C++ programs

## code for quicksort

```
void quicksort ( vector< pair<int,char> >& v )
{
    if(v.size() > 1)
    {
        vector< pair<int,char> > a,b;
        pair<int,char> pivot = v[0];
        for(int i=1; i<v.size(); i++)
            if(v[i].first <= pivot.first)
                a.push_back(v[i]);
            else if(v[i].first > pivot.first)
                b.push_back(v[i]);

        if(a.size() > 0) quicksort(a);
        if(b.size() > 0) quicksort(b);
        for(int i=0; i<a.size(); i++) v[i] = a[i];
        v[a.size()] = pivot;
        for(int i=0; i<b.size(); i++)
            v[i+a.size()+1] = b[i];
    }
}
```

## Set Up

sorting a vector of  
pairs

## Quicksort

the quicksort  
algorithm

C++ code for  
quicksort

## Partition

partitioning a vector  
code for partition

Iterators and  
Timers

partition with iterator  
timing C++ programs

## code for quicksort

## Set Up

sorting a vector of  
pairs

## Quicksort

the quicksort  
algorithm

C++ code for  
quicksort

## Partition

partitioning a vector  
code for partition

Iterators and  
Timers

partition with iterator  
timing C++ programs

```

void quicksort ( vector< pair<int,char> >& v )
{
    if(v.size() > 1)
    {
        vector< pair<int,char> > a,b;
        pair<int,char> pivot = v[0];
        for(int i=1; i<v.size(); i++)
            if(v[i].first <= pivot.first)
                a.push_back(v[i]);
            else if(v[i].first > pivot.first)
                b.push_back(v[i]);

        if(a.size() > 0) quicksort(a);
        if(b.size() > 0) quicksort(b);

        for(int i=0; i<a.size(); i++) v[i] = a[i];
        v[a.size()] = pivot;
        for(int i=0; i<b.size(); i++)
            v[i+a.size()+1] = b[i];
    }
}

```

## code for quicksort

```
void quicksort ( vector< pair<int,char> >& v )
{
    if(v.size() > 1)
    {
        vector< pair<int,char> > a,b;
        pair<int,char> pivot = v[0];
        for(int i=1; i<v.size(); i++)
            if(v[i].first <= pivot.first)
                a.push_back(v[i]);
            else if(v[i].first > pivot.first)
                b.push_back(v[i]);

        if(a.size() > 0) quicksort(a);
        if(b.size() > 0) quicksort(b);

        for(int i=0; i<a.size(); i++) v[i] = a[i];
        v[a.size()] = pivot;
        for(int i=0; i<b.size(); i++)
            v[i+a.size()+1] = b[i];
    }
}
```

## Set Up

sorting a vector of  
pairs

## Quicksort

the quicksort  
algorithm

C++ code for  
quicksort

## Partition

partitioning a vector  
code for partition

Iterators and  
Timers

partition with iterator  
timing C++ programs

## code for quicksort

```
void quicksort ( vector< pair<int,char> >& v )
{
    if(v.size() > 1)
    {
        vector< pair<int,char> > a,b;
        pair<int,char> pivot = v[0];
        for(int i=1; i<v.size(); i++)
            if(v[i].first <= pivot.first)
                a.push_back(v[i]);
            else if(v[i].first > pivot.first)
                b.push_back(v[i]);

        if(a.size() > 0) quicksort(a);
        if(b.size() > 0) quicksort(b);

        for(int i=0; i<a.size(); i++) v[i] = a[i];
        v[a.size()] = pivot;
        for(int i=0; i<b.size(); i++)
            v[i+a.size()+1] = b[i];
    }
}
```

## Set Up

sorting a vector of  
pairs

## Quicksort

the quicksort  
algorithm

C++ code for  
quicksort

## Partition

partitioning a vector  
code for partition

Iterators and  
Timers

partition with iterator  
timing C++ programs

# Cost Analysis

## Set Up

sorting a vector of pairs

## Quicksort

the quicksort algorithm

C++ code for quicksort

## Partition

partitioning a vector  
code for partition

## Iterators and Timers

partition with iterator  
timing C++ programs

In the best case, for  $n$  elements:

- 1 The pivot is at the middle of sequence.
- 2 There are  $\log_2(n)$  levels in recursion.
- 3 Computing the partition costs  $O(n)$ .

$\Rightarrow$  cost of quicksort is  $O(n \log_2(n))$

In the worst case:

- 1 The pivot is always smallest (or largest) element.
- 2 The recursion is  $n$  levels deep.
- 3 Computing the partition costs  $O(n)$ .

$\Rightarrow$  cost of quicksort is  $O(n^2)$

# Cost Analysis

## Set Up

sorting a vector of pairs

## Quicksort

the quicksort algorithm

C++ code for quicksort

## Partition

partitioning a vector  
code for partition

## Iterators and Timers

partition with iterator  
timing C++ programs

In the best case, for  $n$  elements:

- 1 The pivot is at the middle of sequence.
- 2 There are  $\log_2(n)$  levels in recursion.
- 3 Computing the partition costs  $O(n)$ .

$\Rightarrow$  cost of quicksort is  $O(n \log_2(n))$

In the worst case:

- 1 The pivot is always smallest (or largest) element.
- 2 The recursion is  $n$  levels deep.
- 3 Computing the partition costs  $O(n)$ .

$\Rightarrow$  cost of quicksort is  $O(n^2)$

# Cost Analysis

## Set Up

sorting a vector of pairs

## Quicksort

the quicksort algorithm

C++ code for quicksort

## Partition

partitioning a vector  
code for partition

## Iterators and Timers

partition with iterator  
timing C++ programs

In the best case, for  $n$  elements:

- 1 The pivot is at the middle of sequence.
- 2 There are  $\log_2(n)$  levels in recursion.
- 3 Computing the partition costs  $O(n)$ .

$\Rightarrow$  cost of quicksort is  $O(n \log_2(n))$

In the worst case:

- 1 The pivot is always smallest (or largest) element.
- 2 The recursion is  $n$  levels deep.
- 3 Computing the partition costs  $O(n)$ .

$\Rightarrow$  cost of quicksort is  $O(n^2)$

# Cost Analysis

## Set Up

sorting a vector of pairs

## Quicksort

the quicksort algorithm

C++ code for quicksort

## Partition

partitioning a vector  
code for partition

## Iterators and Timers

partition with iterator  
timing C++ programs

In the best case, for  $n$  elements:

- 1 The pivot is at the middle of sequence.
- 2 There are  $\log_2(n)$  levels in recursion.
- 3 Computing the partition costs  $O(n)$ .

$\Rightarrow$  cost of quicksort is  $O(n \log_2(n))$

In the worst case:

- 1 The pivot is always smallest (or largest) element.
- 2 The recursion is  $n$  levels deep.
- 3 Computing the partition costs  $O(n)$ .

$\Rightarrow$  cost of quicksort is  $O(n^2)$

# Cost Analysis

## Set Up

sorting a vector of pairs

## Quicksort

the quicksort algorithm

C++ code for quicksort

## Partition

partitioning a vector  
code for partition

## Iterators and Timers

partition with iterator  
timing C++ programs

In the best case, for  $n$  elements:

- 1 The pivot is at the middle of sequence.
- 2 There are  $\log_2(n)$  levels in recursion.
- 3 Computing the partition costs  $O(n)$ .

$\Rightarrow$  cost of quicksort is  $O(n \log_2(n))$

In the worst case:

- 1 The pivot is always smallest (or largest) element.
- 2 The recursion is  $n$  levels deep.
- 3 Computing the partition costs  $O(n)$ .

$\Rightarrow$  cost of quicksort is  $O(n^2)$

# Cost Analysis

## Set Up

sorting a vector of pairs

## Quicksort

the quicksort algorithm

C++ code for quicksort

## Partition

partitioning a vector  
code for partition

## Iterators and Timers

partition with iterator  
timing C++ programs

In the best case, for  $n$  elements:

- 1 The pivot is at the middle of sequence.
- 2 There are  $\log_2(n)$  levels in recursion.
- 3 Computing the partition costs  $O(n)$ .

$\Rightarrow$  cost of quicksort is  $O(n \log_2(n))$

In the worst case:

- 1 The pivot is always smallest (or largest) element.
- 2 The recursion is  $n$  levels deep.
- 3 Computing the partition costs  $O(n)$ .

$\Rightarrow$  cost of quicksort is  $O(n^2)$

5 Nov 2010

Set Up

sorting a vector of pairs

Quicksort

the quicksort algorithm  
C++ code for quicksort

Partition

partitioning a vector  
code for partition

Iterators and Timers

partition with iterator  
timing C++ programs

# quicksort

- 1 Set Up  
sorting a vector of pairs
- 2 Quicksort  
the quicksort algorithm  
C++ code for quicksort
- 3 Partition  
partitioning a vector  
code for partition
- 4 Iterators and Timers  
partition with iterator  
timing C++ programs

5 Nov 2010

## partitioning a vector

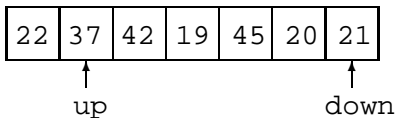
## Set Up

sorting a vector of  
pairs

## Quicksort

the quicksort  
algorithm  
C++ code for  
quicksort

## Partition

partitioning a vector  
code for partitionIterators and  
Timerspartition with iterator  
timing C++ programs

the pivot is 22

up ↑ down ↓

swap if up > pivot  
down < pivot



move up and down  
till ready to swap



continue  
until up == down

5 Nov 2010

## partitioning a vector

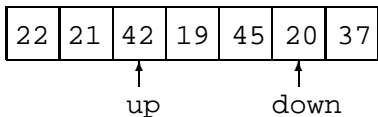
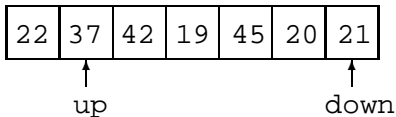
## Set Up

sorting a vector of  
pairs

## Quicksort

the quicksort  
algorithm  
C++ code for  
quicksort

## Partition

partitioning a vector  
code for partitionIterators and  
Timerspartition with iterator  
timing C++ programs

the pivot is 22

up ↑ down ↓

swap if up > pivot  
down < pivotmove up and down  
till ready to swap

continue

until up == down

5 Nov 2010

## partitioning a vector

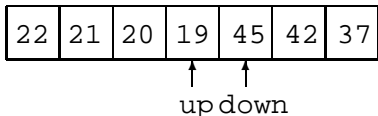
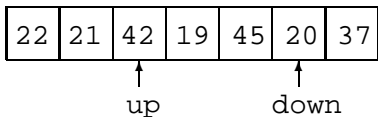
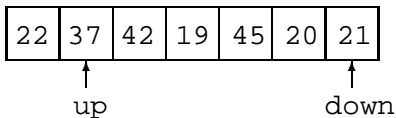
## Set Up

sorting a vector of  
pairs

## Quicksort

the quicksort  
algorithm  
C++ code for  
quicksort

## Partition

partitioning a vector  
code for partitionIterators and  
Timerspartition with iterator  
timing C++ programs

the pivot is 22

up ↑ down ↓

swap if up > pivot  
down < pivotmove up and down  
till ready to swapcontinue  
until up == down

## running an example

## Set Up

sorting a vector of  
pairs

## Quicksort

the quicksort  
algorithm

C++ code for  
quicksort

## Partition

partitioning a vector  
code for partition

Iterators and  
Timers

partition with iterator  
timing C++ programs

7 random items :

(22,e)(37,s)(42,x)(19,u)(45,l)(20,n)(21,i)

swapping 1 with 6

swapping 2 with 5

moving pivot to 3

pivot = 3

the partitioned vector :

-> first half : (19,u)(21,i)(20,n)

-> the pivot : (22,e)

-> second half : (45,l)(42,x)(37,s)

## running an example

## Set Up

sorting a vector of  
pairs

## Quicksort

the quicksort  
algorithm  
C++ code for  
quicksort

## Partition

partitioning a vector  
code for partition

Iterators and  
Timers

partition with iterator  
timing C++ programs

7 random items :

```
(22,e)(37,s)(42,x)(19,u)(45,l)(20,n)(21,i)
```

swapping 1 with 6

```
swapping 2 with 5
```

```
moving pivot to 3
```

```
pivot = 3
```

the partitioned vector :

```
-> first half : (19,u)(21,i)(20,n)
```

```
-> the pivot : (22,e)
```

```
-> second half : (45,l)(42,x)(37,s)
```

## running an example

## Set Up

sorting a vector of  
pairs

## Quicksort

the quicksort  
algorithm  
C++ code for  
quicksort

## Partition

partitioning a vector  
code for partition

Iterators and  
Timers

partition with iterator  
timing C++ programs

```
7 random items :
```

```
(22,e)(37,s)(42,x)(19,u)(45,l)(20,n)(21,i)
```

```
swapping 1 with 6
```

```
swapping 2 with 5
```

```
moving pivot to 3
```

```
pivot = 3
```

```
the partitioned vector :
```

```
-> first half : (19,u)(21,i)(20,n)
```

```
-> the pivot : (22,e)
```

```
-> second half : (45,l)(42,x)(37,s)
```

## running an example

## Set Up

sorting a vector of  
pairs

## Quicksort

the quicksort  
algorithm  
C++ code for  
quicksort

## Partition

partitioning a vector  
code for partition

Iterators and  
Timers

partition with iterator  
timing C++ programs

```
7 random items :
```

```
(22,e)(37,s)(42,x)(19,u)(45,l)(20,n)(21,i)
```

```
swapping 1 with 6
```

```
swapping 2 with 5
```

```
moving pivot to 3
```

```
pivot = 3
```

```
the partitioned vector :
```

```
-> first half : (19,u)(21,i)(20,n)
```

```
-> the pivot : (22,e)
```

```
-> second half : (45,l)(42,x)(37,s)
```

## running an example

## Set Up

sorting a vector of  
pairs

## Quicksort

the quicksort  
algorithm  
C++ code for  
quicksort

## Partition

partitioning a vector  
code for partition

Iterators and  
Timers

partition with iterator  
timing C++ programs

7 random items :

```
(22,e)(37,s)(42,x)(19,u)(45,l)(20,n)(21,i)
```

swapping 1 with 6

swapping 2 with 5

moving pivot to 3

pivot = 3

the partitioned vector :

```
-> first half : (19,u)(21,i)(20,n)
```

```
-> the pivot : (22,e)
```

```
-> second half : (45,l)(42,x)(37,s)
```

## running an example

## Set Up

sorting a vector of  
pairs

## Quicksort

the quicksort  
algorithm  
C++ code for  
quicksort

## Partition

partitioning a vector  
code for partition

Iterators and  
Timers

partition with iterator  
timing C++ programs

7 random items :

```
(22,e)(37,s)(42,x)(19,u)(45,l)(20,n)(21,i)
```

swapping 1 with 6

swapping 2 with 5

moving pivot to 3

pivot = 3

the partitioned vector :

```
-> first half : (19,u)(21,i)(20,n)
```

```
-> the pivot : (22,e)
```

```
-> second half : (45,l)(42,x)(37,s)
```

5 Nov 2010

Set Up

sorting a vector of pairs

Quicksort

the quicksort algorithm  
C++ code for quicksort

Partition

partitioning a vector  
**code for partition**

Iterators and Timers

partition with iterator  
timing C++ programs

# quicksort

- 1 Set Up  
sorting a vector of pairs
- 2 Quicksort  
the quicksort algorithm  
C++ code for quicksort
- 3 Partition  
partitioning a vector  
**code for partition**
- 4 Iterators and Timers  
partition with iterator  
timing C++ programs

## code for partition

```
void partition
( vector< pair<int,char> >& v,
  int lower, int upper, int& pivot )
{
    pair<int,char> x = v[lower];
    int up = lower+1; int down = upper;
    while(up < down)
    {
        while((up < down)
              && (v[up].first <= x.first)) up++;
        while((up < down)
              && (v[down].first > x.first)) down--;
        if(up == down) break;
        pair<int,char> tmp = v[up];
        v[up] = v[down]; v[down] = tmp;
    }
    if(v[up].first > x.first) up--;
    v[lower] = v[up]; v[up] = x; pivot = up;
}
```

## Set Up

sorting a vector of  
pairs

## Quicksort

the quicksort  
algorithm  
C++ code for  
quicksort

## Partition

partitioning a vector  
**code for partition**

Iterators and  
Timers

partition with iterator  
timing C++ programs

## code for partition

## Set Up

sorting a vector of  
pairs

## Quicksort

the quicksort  
algorithm  
C++ code for  
quicksort

## Partition

partitioning a vector  
code for partition

Iterators and  
Timers

partition with iterator  
timing C++ programs

```
void partition
( vector< pair<int,char> >& v,
  int lower, int upper, int& pivot )
{
    pair<int,char> x = v[lower];
    int up = lower+1; int down = upper;
    while(up < down)
    {
        while((up < down)
              && (v[up].first <= x.first)) up++;
        while((up < down)
              && (v[down].first > x.first)) down--;
        if(up == down) break;
        pair<int,char> tmp = v[up];
        v[up] = v[down]; v[down] = tmp;
    }
    if(v[up].first > x.first) up--;
    v[lower] = v[up]; v[up] = x; pivot = up;
}
```

## code for partition

## Set Up

sorting a vector of  
pairs

## Quicksort

the quicksort  
algorithm  
C++ code for  
quicksort

## Partition

partitioning a vector  
code for partition

Iterators and  
Timers

partition with iterator  
timing C++ programs

```

void partition
( vector< pair<int,char> >& v,
  int lower, int upper, int& pivot )
{
    pair<int,char> x = v[lower];
    int up = lower+1; int down = upper;
    while(up < down)
    {
        while((up < down)
              && (v[up].first <= x.first)) up++;
        while((up < down)
              && (v[down].first > x.first)) down--;
        if(up == down) break;
        pair<int,char> tmp = v[up];
        v[up] = v[down]; v[down] = tmp;
    }
    if(v[up].first > x.first) up--;
    v[lower] = v[up]; v[up] = x; pivot = up;
}

```

## code for partition

## Set Up

sorting a vector of  
pairs

## Quicksort

the quicksort  
algorithm  
C++ code for  
quicksort

## Partition

partitioning a vector  
code for partition

Iterators and  
Timers

partition with iterator  
timing C++ programs

```

void partition
( vector< pair<int,char> >& v,
  int lower, int upper, int& pivot )
{
    pair<int,char> x = v[lower];
    int up = lower+1; int down = upper;
    while(up < down)
    {
        while((up < down)
              && (v[up].first <= x.first)) up++;
        while((up < down)
              && (v[down].first > x.first)) down--;
        if(up == down) break;
        pair<int,char> tmp = v[up];
        v[up] = v[down]; v[down] = tmp;
    }
    if(v[up].first > x.first) up--;
    v[lower] = v[up]; v[up] = x; pivot = up;
}

```

## code for partition

## Set Up

sorting a vector of  
pairs

## Quicksort

the quicksort  
algorithm  
C++ code for  
quicksort

## Partition

partitioning a vector  
code for partition

Iterators and  
Timers

partition with iterator  
timing C++ programs

```
void partition
( vector< pair<int,char> >& v,
  int lower, int upper, int& pivot )
{
  pair<int,char> x = v[lower];
  int up = lower+1; int down = upper;
  while(up < down)
  {
    while((up < down)
           && (v[up].first <= x.first)) up++;
    while((up < down)
           && (v[down].first > x.first)) down--;
    if(up == down) break;
    pair<int,char> tmp = v[up];
    v[up] = v[down]; v[down] = tmp;
  }
  if(v[up].first > x.first) up--;
  v[lower] = v[up]; v[up] = x; pivot = up;
}
```

## using partition

## Set Up

sorting a vector of  
pairs

## Quicksort

the quicksort  
algorithm  
C++ code for  
quicksort

## Partition

partitioning a vector  
**code for partition**

Iterators and  
Timers

partition with iterator  
timing C++ programs

```
void quicksort
( vector< pair<int,char> >& v,
  int k, int lb, int ub )
{
  if(ub - lb > 0)
  {
    int pivot;

    partition(v,lb,ub,pivot);

    quicksort(v,k+1,lb,pivot-1);

    quicksort(v,k+1,pivot+1,ub);
  }
}
```

## using partition

## Set Up

sorting a vector of  
pairs

## Quicksort

the quicksort  
algorithm  
C++ code for  
quicksort

## Partition

partitioning a vector  
code for partition

Iterators and  
Timers

partition with iterator  
timing C++ programs

```
void quicksort
( vector< pair<int,char> >& v,
  int k, int lb, int ub )
{
  if(ub - lb > 0)
  {
    int pivot;

    partition(v,lb,ub,pivot);

    quicksort(v,k+1,lb,pivot-1);

    quicksort(v,k+1,pivot+1,ub);
  }
}
```

5 Nov 2010

Set Up

sorting a vector of pairs

Quicksort

the quicksort algorithm  
C++ code for quicksort

Partition

partitioning a vector  
code for partition

Iterators and Timers

partition with iterator  
timing C++ programs

# quicksort

- 1 Set Up  
sorting a vector of pairs
- 2 Quicksort  
the quicksort algorithm  
C++ code for quicksort
- 3 Partition  
partitioning a vector  
code for partition
- 4 Iterators and Timers  
partition with iterator  
timing C++ programs

## writing with iterator

## Set Up

sorting a vector of  
pairs

## Quicksort

the quicksort  
algorithm  
C++ code for  
quicksort

## Partition

partitioning a vector  
code for partition

Iterators and  
Timers

partition with iterator  
timing C++ programs

```
void write_vector
( vector< pair<int,char> >::const_iterator first,
  vector< pair<int,char> >::const_iterator last );
// writes from *first to *last, *last included
```

```
void write_vector
( vector< pair<int,char> >::const_iterator first,
  vector< pair<int,char> >::const_iterator last )
{
    vector< pair<int,char> >::const_iterator i;

    for(i=first; i!=(last+1); i++)
        cout << "(" << i->first
              << "," << i->second
              << ")";
}
```

## writing with iterator

## Set Up

sorting a vector of  
pairs

## Quicksort

the quicksort  
algorithm  
C++ code for  
quicksort

## Partition

partitioning a vector  
code for partition

Iterators and  
Timers

partition with iterator  
timing C++ programs

```
void write_vector
( vector< pair<int,char> >::const_iterator first,
  vector< pair<int,char> >::const_iterator last );
// writes from *first to *last, *last included
```

```
void write_vector
( vector< pair<int,char> >::const_iterator first,
  vector< pair<int,char> >::const_iterator last )
{
    vector< pair<int,char> >::const_iterator i;

    for(i=first; i!=(last+1); i++)
        cout << "(" << i->first
              << "," << i->second
              << ")";
}
```

# partition with iterator

## Set Up

sorting a vector of  
pairs

## Quicksort

the quicksort  
algorithm  
C++ code for  
quicksort

## Partition

partitioning a vector  
code for partition

## Iterators and Timers

partition with iterator  
timing C++ programs

```
vector< pair<int,char> >::iterator partition
( vector< pair<int,char> >::iterator lower,
  vector< pair<int,char> >::iterator upper );

// precondition: lower != upper

// return iterator to pivot equal to *lower
// if pivot = partition(lower,upper) then
// all elements before pivot are <= pivot->first
// and
// all elements after pivot are > pivot->first
// where lower <= pivot <= upper
```

# definition of partition

## Set Up

sorting a vector of  
pairs

## Quicksort

the quicksort  
algorithm  
C++ code for  
quicksort

## Partition

partitioning a vector  
code for partition

## Iterators and Timers

partition with iterator  
timing C++ programs

```
vector< pair<int,char> >::iterator partition
( vector< pair<int,char> >::iterator lower,
  vector< pair<int,char> >::iterator upper )
{
    vector< pair<int,char> >::iterator pivot;
    vector< pair<int,char> >::iterator up = lower+1;
    vector< pair<int,char> >::iterator down = upper;

    pair<int,char> x = *lower;

    while(up != down)
    {
        while((up != down)
              && (up->first <= x.first)) up++;
        while((up != down)
              && (down->first > x.first)) down--;

        if(up == down) break;
    }
}
```

## definition of partition

```

vector< pair<int,char> >::iterator partition
( vector< pair<int,char> >::iterator lower,
  vector< pair<int,char> >::iterator upper )
{
    vector< pair<int,char> >::iterator pivot;
    vector< pair<int,char> >::iterator up = lower+1;
    vector< pair<int,char> >::iterator down = upper;

    pair<int,char> x = *lower;

    while(up != down)
    {
        while((up != down)
              && (up->first <= x.first)) up++;
        while((up != down)
              && (down->first > x.first)) down--;

        if(up == down) break;
    }
}

```

### Set Up

sorting a vector of  
pairs

### Quicksort

the quicksort  
algorithm  
C++ code for  
quicksort

### Partition

partitioning a vector  
code for partition

### Iterators and Timers

partition with iterator  
timing C++ programs

## Set Up

sorting a vector of  
pairs

## Quicksort

the quicksort  
algorithm  
C++ code for  
quicksort

## Partition

partitioning a vector  
code for partition

Iterators and  
Timers

partition with iterator  
timing C++ programs

```
if(up == down) break;
// last line on previous slide copied

pair<int,char> tmp = *up;
*up = *down; *down = tmp;
}

if(up->first > x.first) up--;

*lower = *up;
*up = x;
pivot = up;

return pivot;
}
```

## Set Up

sorting a vector of  
pairs

## Quicksort

the quicksort  
algorithm  
C++ code for  
quicksort

## Partition

partitioning a vector  
code for partition

Iterators and  
Timers

partition with iterator  
timing C++ programs

```
        if(up == down) break;
        // last line on previous slide copied

        pair<int, char> tmp = *up;
        *up = *down; *down = tmp;
    }

    if(up->first > x.first) up--;

    *lower = *up;
    *up = x;
    pivot = up;

    return pivot;
}
```

# quicksort with iterator

## Set Up

sorting a vector of  
pairs

## Quicksort

the quicksort  
algorithm  
C++ code for  
quicksort

## Partition

partitioning a vector  
code for partition

## Iterators and Timers

partition with iterator  
timing C++ programs

```
void quicksort
( vector< pair<int,char> >::iterator lower,
  vector< pair<int,char> >::iterator upper )
{
    if(lower != upper)
    {
        vector< pair<int,char> >::iterator pivot;
        pivot = partition(lower,upper);

        if(lower != pivot) quicksort(lower,pivot-1);
        if(upper != pivot) quicksort(pivot+1,upper);
    }
}
```

# quicksort with iterator

## Set Up

sorting a vector of  
pairs

## Quicksort

the quicksort  
algorithm  
C++ code for  
quicksort

## Partition

partitioning a vector  
code for partition

## Iterators and Timers

partition with iterator  
timing C++ programs

```
void quicksort
( vector< pair<int,char> >::iterator lower,
  vector< pair<int,char> >::iterator upper )
{
    if(lower != upper)
    {
        vector< pair<int,char> >::iterator pivot;
        pivot = partition(lower,upper);

        if(lower != pivot) quicksort(lower,pivot-1);
        if(upper != pivot) quicksort(pivot+1,upper);
    }
}
```

5 Nov 2010

## Set Up

sorting a vector of pairs

## Quicksort

the quicksort algorithm  
C++ code for quicksort

## Partition

partitioning a vector  
code for partition

## Iterators and Timers

partition with iterator  
timing C++ programs

# quicksort

- 1 Set Up  
sorting a vector of pairs
- 2 Quicksort  
the quicksort algorithm  
C++ code for quicksort
- 3 Partition  
partitioning a vector  
code for partition
- 4 Iterators and Timers  
partition with iterator  
timing C++ programs

## timing quicksort

## Set Up

sorting a vector of  
pairs

## Quicksort

the quicksort  
algorithm  
C++ code for  
quicksort

## Partition

partitioning a vector  
code for partition

Iterators and  
Timers

partition with iterator  
timing C++ programs

```
std::clock_t tstart,tstop;

tstart = std::clock();

quicksort(v.begin(),v.end()-1,0);

tstop = std::clock();

cout << "time elapsed : "
      << (tstop - tstart)/double(CLOCKS_PER_SEC)
      << " seconds" << endl;
```

# verify at command line

## Set Up

sorting a vector of  
pairs

## Quicksort

the quicksort  
algorithm  
C++ code for  
quicksort

## Partition

partitioning a vector  
code for partition

## Iterators and Timers

partition with iterator  
timing C++ programs

To verify whether our timer is correct,  
we place the input in the file `/tmp/input`  
and run at the command prompt `$`:

```
$ time /tmp/quicksortv3 < /tmp/input  
give n : time elapsed : 1.20141 seconds
```

```
real 0m1.217s  
user 0m1.213s  
sys 0m0.004s
```

## timing our quicksort

## Set Up

sorting a vector of  
pairs

## Quicksort

the quicksort  
algorithm

C++ code for  
quicksort

## Partition

partitioning a vector  
code for partition

Iterators and  
Timers

partition with iterator  
timing C++ programs

On Mac OS X 10.6.1 at 2.26 Ghz memory 2 GB:

n : 10000      time : 0.021947 seconds

n : 20000      time : 0.070821 seconds

n : 40000      time : 0.258824 seconds

n : 80000      time : 0.984645 seconds

n : 160000     time : 3.85452 seconds

n : 320000     time : 15.0596 seconds

## timing STL sort

## Set Up

sorting a vector of  
pairs

## Quicksort

the quicksort  
algorithm  
C++ code for  
quicksort

## Partition

partitioning a vector  
code for partition

Iterators and  
Timers

partition with iterator  
timing C++ programs

On Mac OS X 10.6.1 at 2.26 Ghz memory 2 GB:

n : 10000      time : 0.004399 seconds

n : 20000      time : 0.008772 seconds

n : 40000      time : 0.017364 seconds

n : 80000      time : 0.035269 seconds

n : 160000     time : 0.075455 seconds

n : 320000     time : 0.160034 seconds

## turn optimizer on

## Set Up

sorting a vector of pairs

## Quicksort

the quicksort algorithm  
C++ code for quicksort

## Partition

partitioning a vector  
code for partition

## Iterators and Timers

partition with iterator  
timing C++ programs

On Mac OS X 10.6.1 at 2.26 Ghz memory 2 GB:

```
g++ -O3 -o /tmp/quicksortv3 quicksortv3.cpp
```

```
n : 10000    time : 0.001357 seconds
```

```
n : 20000    time : 0.003855 seconds
```

```
n : 40000    time : 0.011316 seconds
```

```
n : 80000    time : 0.037179 seconds
```

```
n : 160000   time : 0.136174 seconds
```

```
n : 320000   time : 0.524924 seconds
```

# Summary + Assignments

Ended chapter 10 on sorting algorithms with quicksort.

## Assignments:

- 1 Adjust the quicksort code to count the number of comparisons and swaps. For dimensions  $n = 10, 20, 40, 80, \dots$  run 10 random instances and make a table with the number of comparisons and swaps. Do you notice the  $O(n \log_2(n))$  cost?
- 2 Change the code for partition so that the middle element of the sequence is used as pivot. Demonstrate that your changes work in quicksort.
- 3 Compare quicksort with selection sort on vectors of increasing size, taking timings. For which  $n$  is the effect of  $O(n \log_2(n))$  versus  $O(n^2)$  noticeable?

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