

Quick Sort

David
Casperson

The Ideas

Summary

The End

Quick Sort

David Casperson

¹Department of Computer Science
University of Northern BC

2007-11-06 / CPSC 200 Lecture

Outline

Quick Sort

David
Casperson

The Ideas

Summary

The End

- 1 The Ideas
 - Divide and Conquer
 - Selecting the Pivot
 - Partitioning
 - How to Conquer

- 2 Summary
 - New Ideas
 - Report Card

Quick Sort

Divide and Conquer?

Quick Sort

David
Casperson

The Ideas

Divide and
Conquer

Selecting the
Pivot

Partitioning

How to Conquer

Summary

The End

- We know that we **can** sort in $O(n^2)$ time.
- We know from sub-sequence sum algorithms that if we can split the problem into two half-size problems we win

Quick Sort

Divide and Conquer?

Quick Sort

David
Casperson

The Ideas

Divide and
Conquer

Selecting the
Pivot

Partitioning

How to Conquer

Summary

The End

- We know that we **can** sort in $O(n^2)$ time.
- We know from sub-sequence sum algorithms that if we can split the problem into two half-size problems we win because $2(n/2)^2 < n^2$.
- *Partitioning* splits the problem approximately into half.

Quick Sort

How to Divide

Quick Sort

David
Casperson

The Ideas

Divide and Conquer

Selecting the
Pivot

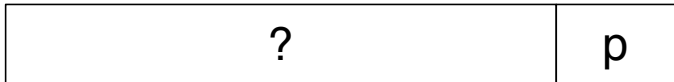
Partitioning

How to Conquer

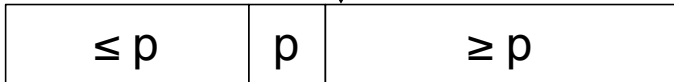
Summary

The End

- The Goal of the Partition algorithm



partition

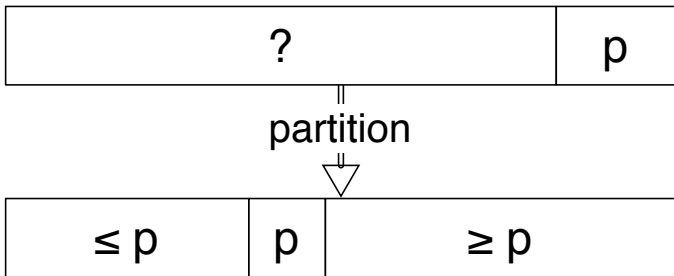


- The problem with the Partition algorithm ...

Quick Sort

How to Divide

- The Goal of the Partition algorithm



- The problem with the Partition algorithm ... we don't know how to choose p .
- Good choices of p lead to $\Theta(n \log n)$
- Bad choices lead to $\Theta(n^2)$.

Quick Sort

David
Casperson

The Ideas

Divide and
Conquer

Selecting the
Pivot

Partitioning

How to Conquer

Summary

The End

The Partition Algorithm

Selecting the pivot

- pivot selection is very important
- *Weiss* uses best of three
- the *Weiss* method puts pivots in place
- the *Weiss* method works well with nearly sorted data

Quick Sort

David
Casperson

The Ideas

Divide and
Conquer

Selecting the
Pivot

Partitioning
How to Conquer

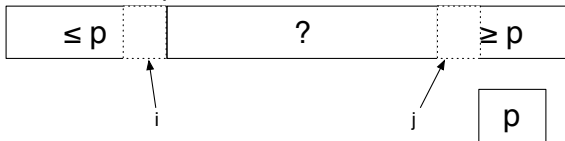
Summary

The End

The Partition Algorithm

The picture

the outer loop



```
while (i<j) {  
    while (data[++i]<p);  
    while (p<data[--j]);  
    if (i<j) std::swap(data[i],data[j]);  
}
```

Quick Sort

David
Casperson

The Ideas

Divide and
Conquer
Selecting the
Pivot

Partitioning
How to Conquer

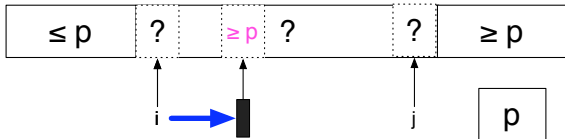
Summary

The End

The Partition Algorithm

The picture

moving i



```
while (i<j) {  
    while (data[++i]<p);  
    while (p<data[--j]);  
    if (i<j) std::swap(data[i],data[j]);  
}
```

Quick Sort

David
Casperson

The Ideas

Divide and
Conquer
Selecting the
Pivot

Partitioning
How to Conquer

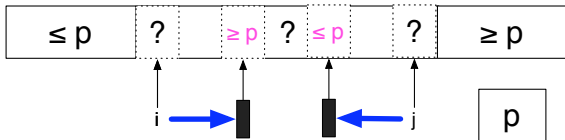
Summary

The End

The Partition Algorithm

The picture

moving j



```
while (i<j) {  
    while (data[++i]<p);  
    while (p<data[--j]);  
    if (i<j) std::swap(data[i],data[j]);  
}
```

Quick Sort

David
Casperson

The Ideas

Divide and
Conquer
Selecting the
Pivot

Partitioning
How to Conquer

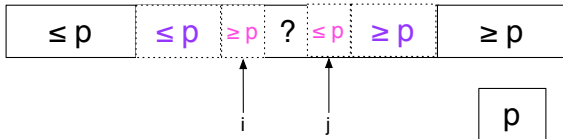
Summary

The End

The Partition Algorithm

The picture

swapping the out of place pair



```
while (i<j) {  
    while (data[++i]<p);  
    while (p<data[--j]);  
    if (i<j) std::swap(data[i],data[j]);  
}
```

Quick Sort

David
Casperson

The Ideas

Divide and
Conquer
Selecting the
Pivot

Partitioning
How to Conquer

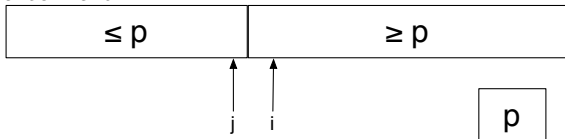
Summary

The End

The Partition Algorithm

The picture

afterward...



```
while (i<j) {  
    while (data[++i]<p);  
    while (p<data[--j]);  
    if (i<j) std::swap(data[i],data[j]);  
}  
.
```

Quick Sort

David
Casperson

The Ideas

Divide and
Conquer

Selecting the
Pivot

Partitioning
How to Conquer

Summary

The End

The Partition Algorithm

The details

- elements equal to the pivot are critical
swap them to make sure split is equal
- sentinels are critical
let pivot selection put them in place
- constants are very good
- exploits modern cache well.

Quick Sort

David
Casperson

The Ideas

Divide and
Conquer

Selecting the
Pivot

Partitioning

How to Conquer

Summary

The End

The overall strategy

code

Quick Sort

David
Casperson

The Ideas

Divide and
Conquer

Selecting the
Pivot

Partitioning
How to Conquer

Summary

The End

```
template <typename elt_t>
void qsort_rec(int b, int e,
               std::vector<elt_t> & data)  {
    if (e-b < cutoff) return ;
    int m = (b+e)/2 ;
    sort3(data[b], data[m], data[e-1]) ;
    std::swap(data[m],data[e-1]) ;
    m = partition(data,b,e-1,data[e-1]) ;
    std::swap(data[m],data[e-1]) ;
    qsort_rec(b,m,data) ;
    qsort_rec(m+1,e,data) ;
    return ;
}
```

Need a base case!

The overall strategy

code

Quick Sort

David
Casperson

The Ideas

Divide and
Conquer

Selecting the
Pivot

Partitioning
How to Conquer

Summary

The End

```
template <typename elt_t>
void qsort_rec(int b, int e,
               std::vector<elt_t> & data)  {
    if (e-b < cutoff) return ;
    int m = (b+e)/2 ;
    sort3(data[b], data[m], data[e-1]) ;
    std::swap(data[m],data[e-1]) ;
    m = partition(data,b,e-1,data[e-1]) ;
    std::swap(data[m],data[e-1]) ;
    qsort_rec(b,m,data) ;
    qsort_rec(m+1,e,data) ;
    return ;
}
```

Pick a pivot and move it out of the way.



The overall strategy

code

Quick Sort

David
Casperson

The Ideas

Divide and
Conquer

Selecting the
Pivot

Partitioning
How to Conquer

Summary

The End

```
template <typename elt_t>
void qsort_rec(int b, int e,
               std::vector<elt_t> & data)  {
    if (e-b < cutoff) return ;
    int m = (b+e)/2 ;
    sort3(data[b], data[m], data[e-1]) ;
    std::swap(data[m],data[e-1]) ;
    m = partition(data,b,e-1,data[e-1]) ;
    std::swap(data[m],data[e-1]) ;
    qsort_rec(b,m,data) ;
    qsort_rec(m+1,e,data) ;
    return ;
}
```

Partition and move the pivot back.

The overall strategy

code

Quick Sort

David
Casperson

The Ideas

Divide and
Conquer

Selecting the
Pivot

Partitioning

How to Conquer

Summary

The End

```
template <typename elt_t>
void qsort_rec(int b, int e,
               std::vector<elt_t> & data)  {
    if (e-b < cutoff) return ;
    int m = (b+e)/2 ;
    sort3(data[b], data[m], data[e-1]) ;
    std::swap(data[m],data[e-1]) ;
    m = partition(data,b,e-1,data[e-1]) ;
    std::swap(data[m],data[e-1]) ;
    qsort_rec(b,m,data) ;
    qsort_rec(m+1,e,data) ;
    return ;
}
```

Solve one subproblem.

The overall strategy

code

Quick Sort

David
Casperson

The Ideas

Divide and
Conquer

Selecting the
Pivot

Partitioning

How to Conquer

Summary

The End

```
template <typename elt_t>
void qsort_rec(int b, int e,
               std::vector<elt_t> & data)  {
    if (e-b < cutoff) return ;
    int m = (b+e)/2 ;
    sort3(data[b], data[m], data[e-1]) ;
    std::swap(data[m],data[e-1]) ;
    m = partition(data,b,e-1,data[e-1]) ;
    std::swap(data[m],data[e-1]) ;
    qsort_rec(b,m,data) ;
    qsort_rec(m+1,e,data) ;
    return ;
}
```

Solve the other subproblem.

The overall strategy

code

Quick Sort

David
Casperson

The Ideas

Divide and
Conquer

Selecting the
Pivot

Partitioning
How to Conquer

Summary

The End

```
template <typename elt_t>
void qsort_rec(int b, int e,
               std::vector<elt_t> & data)  {
    if (e-b < cutoff) return ;
    int m = (b+e)/2 ;
    sort3(data[b], data[m], data[e-1]) ;
    std::swap(data[m],data[e-1]) ;
    m = partition(data,b,e-1,data[e-1]) ;
    std::swap(data[m],data[e-1]) ;
    qsort_rec(b,m,data) ;
    qsort_rec(m+1,e,data) ;
    return ;
}
```

Done.

The Overall Strategy

managing space

Quick Sort

David
Casperson

The Ideas

Divide and
Conquer

Selecting the
Pivot

Partitioning

How to Conquer

Summary

The End

To manage space

- solve the smaller problem first
- solve the larger problem non-recursively
- This leads to a stack depth of at most $\lceil \log_2 n \rceil$

The Overall Strategy

managing space

Quick Sort

David
Casperson

The Ideas

Divide and
Conquer

Selecting the
Pivot

Partitioning

How to Conquer

Summary

The End

```
template <typename elt_t>
void qsort_rec(int b, int e,
               std::vector<elt_t> & data)
{
    if (e-b < cutoff) return;

    int m = (b+e)/2 ;
    sort3(data[b], data[m], data[e-1]) ;
    std::swap(data[m],data[e-1]) ;
    m = partition(data,b,e-1,data[e-1]) ;
    std::swap(data[m],data[e-1]) ;
    if(2*m>b+e) { qsort_rec(m+1,e,data) ; e=m ;}
    else      { qsort_rec(b,m,data) ; b=m+1;}
}
```

The Overall Strategy

managing space

Quick Sort

David
Casperson

The Ideas

Divide and
Conquer

Selecting the
Pivot

Partitioning
How to Conquer

Summary

The End

```
template <typename elt_t>
void qsort_rec(int b, int e,
               std::vector<elt_t> & data)
{
    while (!(e-b < cutoff)) {
        int m = (b+e)/2 ;
        sort3(data[b], data[m], data[e-1]) ;
        std::swap(data[m],data[e-1]) ;
        m = partition(data,b,e-1,data[e-1]) ;
        std::swap(data[m],data[e-1]) ;
        if(2*m>b+e) { qsort_rec(m+1,e,data) ; e=m ;}
        else      { qsort_rec(b,m,data) ; b=m+1;}
    }
    return ;
}
```

The Overall Strategy

managing space

Quick Sort

David
Casperson

The Ideas

Divide and
Conquer

Selecting the
Pivot

Partitioning
How to Conquer

Summary

The End

```
template <typename elt_t>
void qsort_rec(int b, int e,
               std::vector<elt_t> & data)
{
    while (!(e-b < cutoff)) {
        int m = (b+e)/2 ;
        sort3(data[b], data[m], data[e-1]) ;
        std::swap(data[m],data[e-1]) ;
        m = partition(data,b,e-1,data[e-1]) ;
        std::swap(data[m],data[e-1]) ;
        if(2*m>b+e) { qsort_rec(m+1,e,data) ; e=m ;}
        else      { qsort_rec(b,m,data) ; b=m+1;}
    }
    return ;
}
```

The Overall Strategy

managing space

Quick Sort

David
Casperson

The Ideas

Divide and
Conquer

Selecting the
Pivot

Partitioning

How to Conquer

Summary

The End

```
template <typename elt_t>
void qsort_rec(int b, int e,
               std::vector<elt_t> & data)
{
    while (!(e-b < cutoff)) {
        int m = (b+e)/2 ;
        sort3(data[b], data[m], data[e-1]) ;
        std::swap(data[m],data[e-1]) ;
        m = partition(data,b,e-1,data[e-1]) ;
        std::swap(data[m],data[e-1]) ;
        if(2*m>b+e) { qsort_rec(m+1,e,data) ; e=m ;}
        else      { qsort_rec(b,m,data) ; b=m+1;}
    }
    return ;
}
```


The Overall Strategy

Avoiding n^2 time

Quick Sort

David
Casperson

To manage time

The Ideas

Divide and
Conquer

Selecting the
Pivot

Partitioning

How to Conquer

Summary

The End

The Overall Strategy

Avoiding n^2 time

Quick Sort

David
Casperson

The Ideas

Divide and
Conquer

Selecting the
Pivot

Partitioning

How to Conquer

Summary

The End

To manage time

- abandon ship when we clearly aren't making progress.
- if we are more than about 40 problems deep, something has gone wrong.

Quick Sort Summary

New Ideas

Quick Sort

David
Casperson

The Ideas

Summary
New Ideas
Report Card

The End

Quick sort:

- uses partitioning
- uses sentinels
- works well with hardware
- requires care

Quick Sort Summary

Report Card

Quick Sort

David
Casperson

The Ideas

Summary

New Ideas

Report Card

The End

- $T_{\text{worst}}(n) = \Theta(n^2)$

Quick Sort Summary

Report Card

Quick Sort

David
Casperson

The Ideas

Summary

New Ideas

Report Card

The End

- $T_{worst}(n) = \Theta(n^2)$
happens when continual bad partitions happen

Quick Sort Summary

Report Card

Quick Sort

David
Casperson

The Ideas

Summary

New Ideas

Report Card

The End

- $T_{worst}(n) = \Theta(n^2)$
happens when continual bad partitions happen
- $T_{ave}(n) = \Theta(n \log n)$
almost all the time

Quick Sort Summary

Report Card

Quick Sort

David
Casperson

The Ideas

Summary

New Ideas

Report Card

The End

- $T_{worst}(n) = \Theta(n^2)$
happens when continual bad partitions happen
- $T_{ave}(n) = \Theta(n \log n)$
almost all the time
- $O(\log n)$ extra storage for recursion,

Quick Sort Summary

Report Card

Quick Sort

David
Casperson

The Ideas

Summary

New Ideas
Report Card

The End

- $T_{worst}(n) = \Theta(n^2)$
happens when continual bad partitions happen
- $T_{ave}(n) = \Theta(n \log n)$
almost all the time
- $O(\log n)$ extra storage for recursion, **with care.**

Quick Sort Summary

Report Card

Quick Sort

David
Casperson

The Ideas

Summary

New Ideas
Report Card

The End

- $T_{worst}(n) = \Theta(n^2)$
happens when continual bad partitions happen
- $T_{ave}(n) = \Theta(n \log n)$
almost all the time
- $O(\log n)$ extra storage for recursion, with care.

Quick Sort Summary

Report Card

Quick Sort

David
Casperson

The Ideas

Summary

New Ideas
Report Card

The End

- $T_{worst}(n) = \Theta(n^2)$
happens when continual bad partitions happen
- $T_{ave}(n) = \Theta(n \log n)$
almost all the time
- $O(\log n)$ extra storage for recursion, with care.
- **not** stable.
- very fast

The End

Quick Sort

David
Casperson

The Ideas

Summary

The End