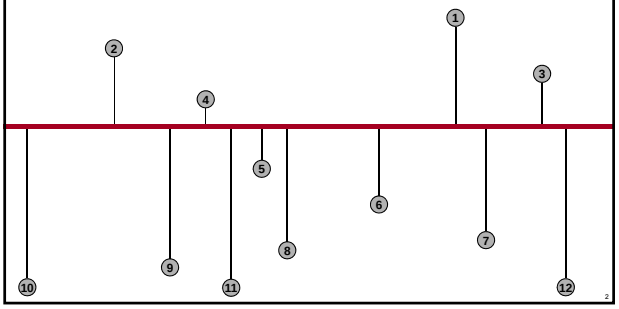
算法设计与分析
Algorithms Design & Analysis

第七讲：统计算法

1

引言：修路问题（Where to Build a Road?）

Given (x,y) coordinates of N houses, where should you build road parallel to x-axis to minimize construction cost of building driveways?

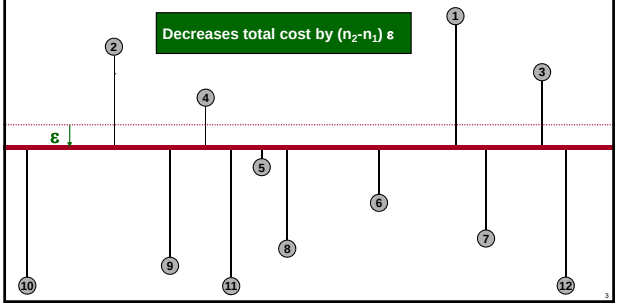


2

Where to Build a Road?

Given (x,y) coordinates of N houses, where should you build road parallel to x-axis to minimize construction cost of building driveways?

- n_1 = nodes on top.（上面的节点数目）
- n_2 = nodes on bottom.（下面的节点数目）

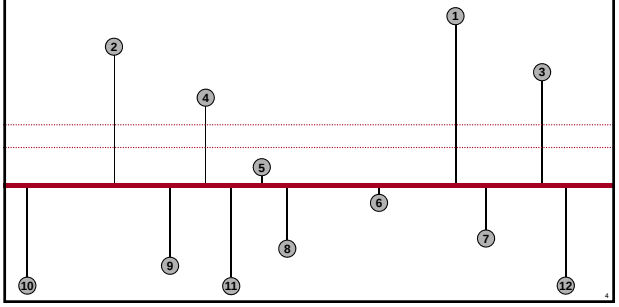


3

Where to Build a Road?

Given (x,y) coordinates of N houses, where should you build road parallel to x-axis to minimize construction cost of building driveways?

- n_1 = nodes on top.
- n_2 = nodes on bottom.

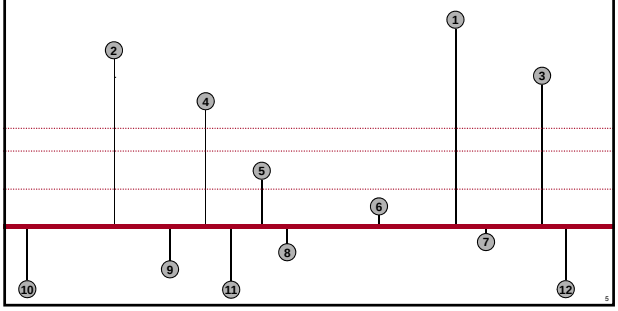


4

Where to Build a Road?

Given (x,y) coordinates of N houses, where should you build road parallel to x-axis to minimize construction cost of building driveways?

- n_1 = nodes on top.
- n_2 = nodes on bottom.

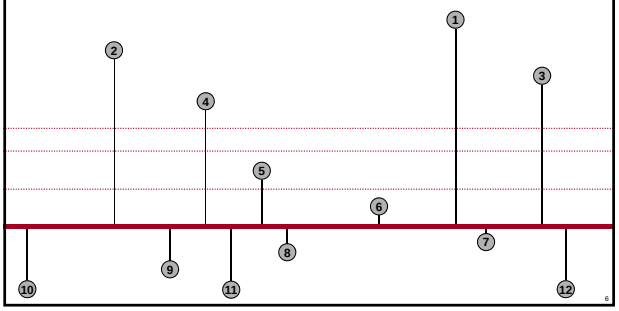


5

Where to Build a Road?

Given (x,y) coordinates of N houses, where should you build road parallel to x-axis to minimize construction cost of building driveways?

Solution: put street at median of y coordinates.（答案）



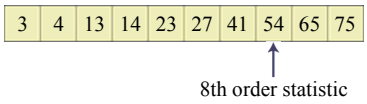
6

序列统计(Order Statistics)

- Finding the minimum and maximum
- (求极小和极大)
- Finding the k-th smallest element
- (求第k小元素)

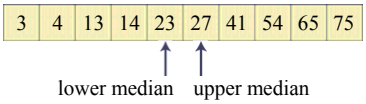
序列统计(Order Statistics)

The ***k-th order statistic*** is the *k*-th smallest element of an array.
(数组中的第*k*小元素)



中位数(Median)

The ***lower median*** is the $\lfloor (n+1)/2 \rfloor$-th order statistic(下中位)
The ***upper median*** is the $\lceil (n+1)/2 \rceil$-th order statistic(上中位)
If *n* is odd, lower and upper median are the same(奇数)



求极小或极大(Finding the Minimum or Maximum)

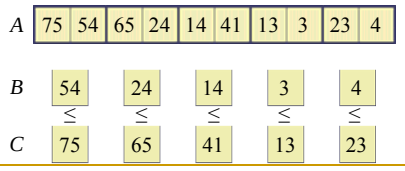
- **Minimum(*A*)**
- 1 *min* ← *A*[1]
- 2 **for** *i* = 2..*n*
- 3 **do if** *A*[*i*] < *min*
- 4 **then** *min* ← *A*[*i*]
- 5 **return** *min*

Lemma: ***The minimum or maximum of a set of n elements can be found using n – 1 comparisons.(n-1次比较)***

同时求极大极小(Finding Both the Minimum and the Maximum)

- **The easy answer(简单答案):**
- *The minimum and the maximum of n elements can be found using 2n – 2 comparisons.(2n – 2次比较)*
- But can we do better, say using *n* comparisons or 3*n* / 2?(能否更快? 3*n* / 2次? *n*次?)

```
Min-Max(A)
1  for i = 1..n / 2
2      do if A[2i – 1] ≤ A[2i]
3          then B[i] ← A[2i – 1]
4               C[i] ← A[2i]
5          else B[i] ← A[2i]
6               C[i] ← A[2i – 1]
7  min ← Minimum(B)
8  max ← Maximum(C)
9  return (min, max)
```



Min-Max(*A*)

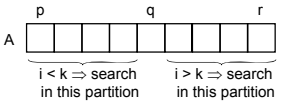
```
1  for i = 1..n / 2
2      do if A[2i – 1] ≤ A[2i]
3          then B[i] ← A[2i – 1]
4               C[i] ← A[2i]
5          else B[i] ← A[2i]
6               C[i] ← A[2i – 1]
7  min ← Minimum(B)
8  max ← Maximum(C)
9  return (min, max)
```

- **Number of Comparisons:**
- *n* / 2 to construct arrays *B* and *C* from array *A*(*n* / 2*次得到B和C*)
- *n* / 2 – 1 to find the minimum in *B*(*从B中找到最小*)
- *n* / 2 – 1 to find the maximum in *C*(*从C中找到最大*)
- **Total: 3*n* / 2 – 2(总共)**

Lemma: ***The minimum and maximum of a set of n numbers can be found using ⌊3n / 2 – 2⌋ comparisons.(比较次数)***

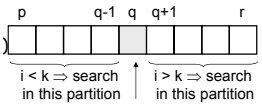
选择问题(General Selection Problem)

- Select the i-th order statistic (i-th smallest element) form a set of n distinct numbers(从序列中选出第i小元素)



- Idea:
  - Partition the input array similarly with the approach used for Quicksort (use RANDOMIZED-PARTITION) (用快速排序中的分割算法对序列进行分割)
  - Recurse on one side of the partition to look for the i-th element depending on where i is with respect to the pivot(在分割后的一边寻找第i小元素,需要与分割中心位置比较)
- Selection of the i-th smallest element of the array *A* can be done in Θ(n) time (可以以Θ(n)开销完成)

随机选择算法(Randomized Select)

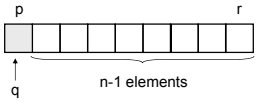


- *Alg.:* RANDOMIZED-SELECT(*A*, p, r, i)
- **if** p = r
- **then return** *A*[p]
- q ←RANDOMIZED-PARTITION(*A*, p, r) (返回值)
- k ← q - p + 1 (分割中心所在位置)
- **if** i = k ▷pivot value is the answer
- **then return** *A*[q]
- **elseif** i < k
- **then return** RANDOMIZED-SELECT(*A*, p, q-1, i)
- **else return** RANDOMIZED-SELECT(*A*, q + 1, r, i-k)

分析(Analysis of Running Time)

- **Worst case** running time: Θ(n^2) (最坏情况)
  - If we always partition around the largest/smallest remaining element(以最大/最小元素为分割中心)
  - Partition takes Θ(n) time (分割开销: Θ(n))
  - T(n) = O(1) (choose the pivot) + Θ(n) (partition) + T(n-1)

  = 1 + n + T(n-1) = Θ(n^2) (总的开销)



分析(Analysis of Running Time)

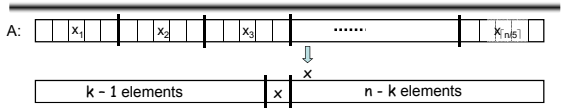
- **Expected** running time (on **average**) (平均情况)

$$E[T(n)] \le \frac{2}{n} \sum_{k=\lfloor n/2 \rfloor}^{n-1} [T(k)] + O(n)$$

更好的选择算法（A Better Selection Algorithm）

- Can perform Selection in *O*(n) Worst Case （在最坏情况下选择开销为*O*(n) ）
- Idea: guarantee a good split on partitioning（确保获得好的分割结果）
  - Running time is influenced by how “balanced” are the resulting partitions（运行时间受到分割结果的均衡性影响）
- Use a modified version of PARTITION （采用改进的分割方法，不再是以序列的最后一个元素为分割中心）

Selection in $O(n)$ Worst Case



1. Divide the n elements into groups of 5 $\Rightarrow \lceil n/5 \rceil$ groups （分成5组）
2. Find the median of each of the $\lceil n/5 \rceil$ groups （找到每组的中值）
   - Use insertion sort, then pick the median
3. Use SELECT recursively to find the median x of the $\lceil n/5 \rceil$ medians （找到5组中值的中值x）
4. Partition the input array around x, using the modified version of PARTITION（运用改进的分割算法在x处分割）
   - There are k-1 elements on the low side of the partition and n-k on the high side（左： k-1；右： n-k）
5. If $i = k$ then return x. Otherwise, use SELECT recursively: 如果$i = k$，返回x结束；否则，在左右递归求解）
   - Find the i-th smallest element on the low side if $i < k$
   - Find the $(i-k)$-th smallest element on the high side if $i > k$

19

Example

- Find the –6th smallest element in array:
- A = {12, 34, 0, 3, 22, 4, 17, 32, 3, 28, 43, 82, 25, 27, 34, 2 ,19 ,12 ,5 ,18 ,20 ,33, 16, 33, 21, 30, 3, 47}

1. Divide the array into groups of 5 elements （分成5组）

12	4	43	2	20	30
34	17	82	19	33	3
0	32	25	12	16	47
3	3	27	5	33	
22	28	34	18	21	

20

Example (cont.)

2. Sort the groups and find their medians （找到每组的中值）

0	4	25	2	20	3
3	3	27	5	16	30
12	17	34	12	21	47
34	32	43	19	33	
22	28	82	18	33	

3. Find the median of the medians （找到5组中值的中值x）

- 12, 12, 17, 21, 34, 30

21

Example (cont.)

4. Partition the array around the median of medians (17) （在5组中值的中值17处分割，三个部分如下）

- First partition:
- {12, 0, 3, 4, 3, 2, 12, 5, 16, 3}
- Pivot:
- 17 (position of the pivot is q = 11)
- Second partition:
- {34, 22, 32, 28, 43, 82, 25, 27, 34, 19, 18, 20, 33, 33, 21, 30, 47}
- To find the 6-th smallest element we would have to recurse our search in the first partition. （在第一部分递归求解）

22

Recurrence for the Running Time

- Step 1: making groups of 5 elements takes $O(n)$ time
- Step 2: sorting $n/5$ groups in $O(1)$ time each takes $O(n)$
- Step 3: calling SELECT on $\lceil n/5 \rceil$ medians takes time $T(\lceil n/5 \rceil)$
- Step 4: partitioning the n-element array around x takes $O(n)$ time
- Step 5: recursing on one partition takes time $\le T(7n/10 + 6)$ 阅读教材
- $T(n) = T(\lceil n/5 \rceil) + T(7n/10 + 6) + O(n)$
- Show that $T(n) = O(n)$

23