Randomization in Algorithms and Data Structures

3 lectures (Tuesdays 14:15-16:00)
- May 1: Gerth Stølting Brodal
- May 8: Kasper Green Larsen
- May 15: Peyman Afshani

For each lecture one handin exercise – deadline June 1
Backwards Analysis

Gerth Stølting Brodal
Pareto optimal / non-dominating points / skyline

Vilfredo Pareto (1848–1923)
Exercise 1 (skyline construction)

Given \( n \) points \( (x_1, y_1), \ldots, (x_n, y_n) \) in \( \mathbb{R}^2 \)

- Give an algorithm for computing the points on the skyline?
- What is the running time of your algorithm?
Problem – expected skyline size?

- Consider $n$ points $(x_1, y_1), \ldots, (x_n, y_n)$ in $R^2$
- Each $x_i$ and $y_i$ is selected independently and uniformly at random from $[0, 1]$
- What is the expected skyline size?
Exercise 2 (dependent points)

- Describe an algorithm for generating \( n \) distinct points \((x_1, y_1), \ldots, (x_n, y_n)\) in \( R^2 \)
- Each \( x_i \) and \( y_i \) is selected uniformly at random from \([0, 1] \)
- The points are not independent
Generating random points

Assume the points are generated by the following algorithm

1) Generate $n$ random $x$-values $x_1, \ldots, x_n$
2) sort the $x$-values in decreasing order
3) for decreasing $x_i$ generate random $y_i$

$(x_i, y_i)$ is a skyline point $\iff y_i = \max(y_1, \ldots, y_i)$

$$\Pr[y_i \text{ skyline point}] = \frac{1}{i}$$

since $y_1, \ldots, y_i$ independent and the same distribution, all permutations are equally likely, i.e. probability $y_i$ to be largest among $i$ values is $1/i$
Expected skyline size

Stochastic variable:  \( X_i = \begin{cases} 1 & \text{if point } i \text{ on skyline} \\ 0 & \text{otherwise} \end{cases} \)

\[
E[|\text{skyline}|] = E[X_1 + \cdots + X_n] = \frac{1}{1} + \frac{1}{2} + \cdots + \frac{1}{n} = \sum_{i=1..n} \frac{1}{i} \quad \text{(harmonic number } H_n) \\
\leq \ln n + 1
\]
The $n$-th Harmonic number $H_n = 1/1 + 1/2 + 1/3 + \cdots + 1/n = \sum_{i=1}^{n} 1/i$

$$H_n - 1 \leq \int_{1}^{n} \frac{1}{x} \, dx = \left[ \ln x \right]_{1}^{n} = \ln n - \ln 1 = \ln n \leq H_n - 1/n$$

$$\ln n + 1/n \leq H_n \leq \ln n + 1$$
Exercise 3 (divide-and-conquer skyline)

Consider the following algorithm

- Find the topmost point \( p \) in \( O(n) \) time
- recurse on points to the right of \( p \)

Show that the expected running time is \( O(n) \)
QuickSort – a randomized sorting algorithm

QuickSort($x_1, ..., x_n$)
- Pick a random pivot element $x_i$
- Partition remaining elements: $S$ smaller than $x_i$, and $L$ larger than $x_i$
- Recursively sort $S$ and $L$
QuickSort – analysis I

Alternative Quicksort

- Consider a random permutation $\pi$ of the input, such that $x_{\pi(1)}$ is the first pivot, then $x_{\pi(2)}$, $x_{\pi(3)}$, ....

Observations

- Changes the order unsorted sublists are partitioned, but the pivots are still selected uniformly among the elements
- $x_i$ and $x_j$ are compared if and only $x_i$ or $x_j$ is selected as a pivot before any element in the sorted list between $x_i$ and $x_j$
QuickSort – analysis II

\[ E[\text{comparisons by quicksort}] = \Sigma_{i<j} \text{cost of comparing } x_i \text{ and } x_j \]

\[ = \Sigma_{i<j} E[\text{cost of comparing } x_i \text{ and } x_j] \]

\[ = \Sigma_{i<j} \frac{2}{|r(j) - r(i)| + 1} \quad \text{where } r(i) = \text{position of } x_i \text{ in output} \]

\[ = \Sigma_{1\leq p < q \leq n} \frac{2}{q - p + 1} \]

\[ \leq 2n \cdot \Sigma_{2\leq d \leq n} \frac{1}{d} \]

\[ \leq 2n \cdot ((\ln n + 1) - 1) \]

\[ = 2n \cdot \ln n \]
Exercise 4 (random search trees)

Construct a unbalanced binary search tree for \( n \) numbers \( x_1 < \cdots < x_n \) by inserting the numbers in random order

- What is the probability that \( x_j \) is an ancestor of \( x_i \)?
- What is the expected depth of a node \( x_i \)?

Insert: 15, 8, 17, 13, 3, 5, 10
Convex hull

Convex hull = smallest polygon enclosing all points
Exercise 5: Convex hull

Give an $O(n \cdot \log n)$ worst-case algorithm finding the Convex Hull
Convex hull – randomized incremental

1) Let $p_1, \ldots, p_n$ be a random permutation of the points
2) Compute convex hull of $\{p_1, p_2, p_3\}$
3) $c = (p_1 + p_2 + p_3) / 3$
4) for $i = 4$ to $n$ insert $p_i$ and construct $H_i$ from $H_{i-1}$:
   if $p_i$ inside $H_{i-1}$ skip, otherwise insert $p_i$ in $H_{i-1}$ and delete chain inside

Convex hull
Convex hull – inserting $p_i$
Convex hull – inserting $p_i$

How to find $e$ for $p_i$?

store set of points with $e$
and reference to $e$ from $p_i$
Convex hull - analysis

- Each point inserted / deleted / inside at most once in convex hull
- $E[\# \text{ point-edge updates}]$
  
  $= E[\sum_{4 \leq i \leq n} \sum p \ p \text{ updated on insertion } i]$
  
  $= \sum_{4 \leq i \leq n} \sum p \ E[p \text{ updated on insertion } i]$
  
  $\leq \sum_{4 \leq i \leq n} \sum p \frac{2}{i - 3}$
  
  $\leq 2n \cdot (\ln n + 1)$

  since $p$ only updated on insertion $i$ if $p_i$ is $u$ or $v$

- Total expected time $O(n \cdot \log n)$
Binary search - but forgot to sort the array... (a debugging case)

How many cells can ever be reached by a binary search?

Binary searching unsorted array
Reachable nodes – analysis

$$\Pr[ v_i \text{ useful } ] = \frac{|L_i|}{\Sigma_j |L_j|}$$

$$E[ \# \text{ useful nodes at level } ] = \Sigma_i \left( \frac{|L_i|}{\Sigma_j |L_j|} \right) = 1$$

$$E[ \# \text{ useful nodes in tree } ] = \text{height} - 1$$

$$E[ \# \text{ reachable nodes in tree } ] \leq \text{height}^2 = O(\log^2 n)$$

Binary searching unsorted array
Closest pair

Given $n$ points, find pair $(p, q)$ with minimum distance

Algorithm (idea)

- permute points randomly $\rightarrow p_1, p_2, \ldots, p_n$
- for $i = 2..n$ compute $\Delta_i = $ distance between closest pair for $p_1, \ldots, p_i$

Observation

- $\Pr[\Delta_i < \Delta_{i-1}] \leq \frac{2}{i}$ since minimum distance can only decrease if $p_i$ is defining $\Delta_i$
Closest pair – grid cells

- Construct grid cells with side-length $\Delta_{i-1}$
- Point $(x, y)$ in cell $([x/\Delta_{i-1}], [y/\Delta_{i-1}])$
- $\leq 4$ points in cell if all pairwise distances $\geq \Delta_{i-1}$
- Neighbors of $p_i$ within distance $\Delta_{i-1}$ are in $\leq 9$ cells
- Store non-empty cells in a hash table (using randomization)
- Rebuild grid whenever $\Delta_i$ decreases

Analysis

- $E[\text{rebuild cost}] = E[\Sigma_{3 \leq i \leq n} \text{rebuild cost inserting } p_i]$
  - $= \Sigma_{3 \leq i \leq n} E[\text{rebuild cost inserting } p_i] \leq \Sigma_{3 \leq i \leq n} 2i \leq 2n$
- Total expected time $O(n)$
Handin (Treaps)

A treap is a binary search tree with a random priority assigned to each element when inserted (in the example elements are white and priorities yellow).

A left-to-right inorder traversal gives the elements in sorted order, whereas the priorities satisfy heap order, i.e. priorities increase along a leaf-to-root path.

a) Argue that an arbitrary element in a treap of size $n$ has expected depth $O(\log n)$

b) Describe how to insert an element into a treap and give running time
References


