Introduction to Algorithms

Mong-Jen Kao (高孟駿)

Tuesday 10:10 – 12:00

Thursday 15:30 – 16:20

Program Assignment - II

Quicksort Algorithm

- Quicksort is a powerful sorting algorithm.
  - It partitions the input into two buckets by a chosen pivot.
  - Then it sorts the two buckets using recursive quicksort.
- If the input is always evenly partitioned,
  quicksort runs in $O(n \log n)$ time.
- In the worst case, however, quicksort takes $O(n^2)$ time.

Randomized Quicksort

- In the randomized Quicksort algorithm,
  we use a randomly chosen pivot to partition the input.
  - This avoids the worst-case most of the times.

- Let $T(n)$ be the expected number of comparisons made by this algorithm. Then, we have

$$T(n) = (n-1) + \frac{1}{n} \cdot \sum_{0 \le i < n} \left(T(i) + T(n-i+1) \right) .$$

- Use a randomly chosen pivot to partition the input.
- Let $T(n)$ be the expected number of comparisons made by this algorithm. Then, we have

$$T(n) = (n-1) + \frac{1}{n} \cdot \sum_{0 \le i < n} \big(T(i) + T(n-i+1)\big)$$

$$= (n-1) + \frac{2}{n} \cdot \sum_{1 \le i < n} T(i) \ .$$

- To solve the recurrence, we guess that $T(n) = O(n \log n)$.

- Let $T(n)$ be the expected number of comparisons made by this algorithm. Then, we have

$$T(n) = (n-1) + \frac{2}{n} \cdot \sum_{1 \le i < n} T(i) \ .$$

- To solve the recurrence, we guess that $T(n) = O(n \log n)$.

Then,

$$T(n) \le (n-1) + \frac{2}{n} \cdot \sum_{1 \le i < n} cn \log n$$

$$\le (n-1) + \frac{2}{n} \cdot \int_1^n cx \log x \cdot dx \le cn \log n \ .$$

For boundary condition, we have $T(1) = 0$.

holds when $c \ge 2$.

Alternative Analysis

- Suppose that the input numbers after sorted are

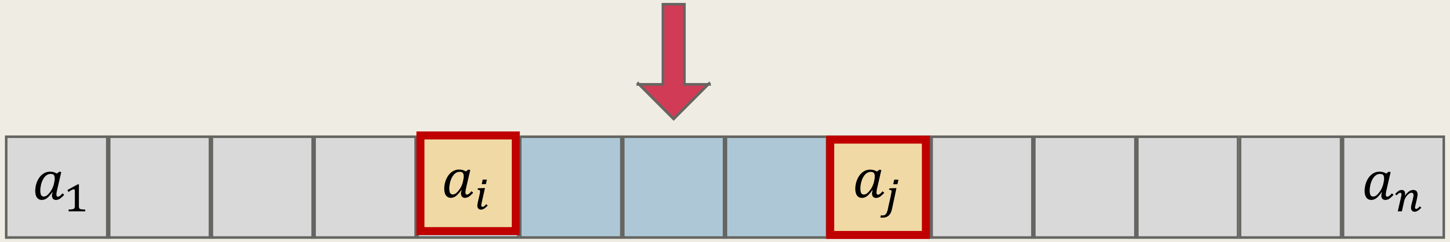
$$a_1 < a_2 < \cdots < a_n \,.$$

- Let X be the ***total number of comparisons*** made by the randomized quicksort algorithm.

- For any $1 \leq i < j \leq n$, let $\boldsymbol{X_{i,j}}$ denote the ***indicator variable*** for the event that ***$\boldsymbol{a_i}$ and $\boldsymbol{a_j}$ are compared*** during execution.

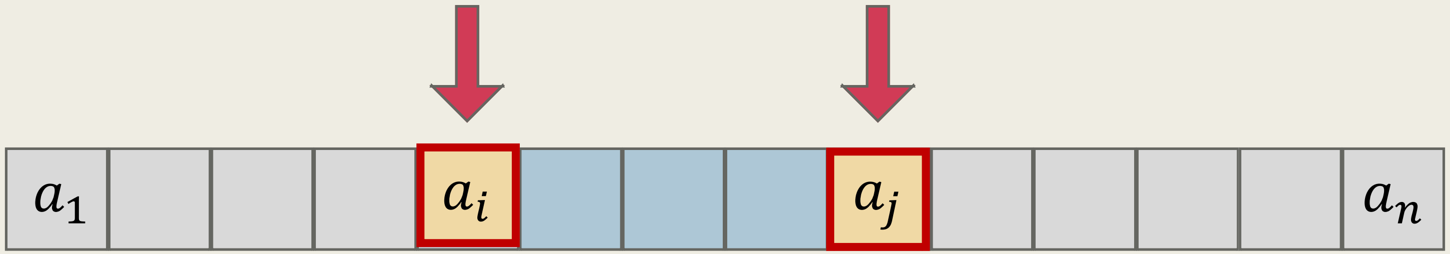
- Then $E[X] = \sum_{i,j} E\left[X_{i,j}\right] = \sum_{i,j} \Pr\left[X_{i,j} = 1\right]$.

When does a_i and a_j get compared?



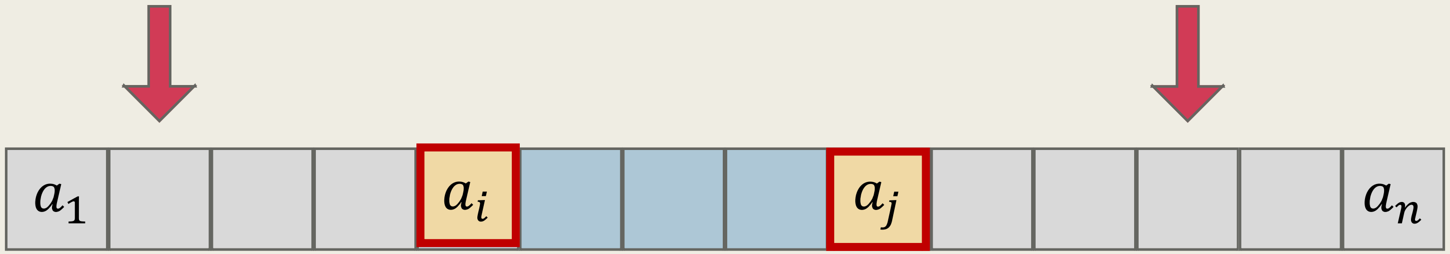
- If some number between a_i and a_j is picked as pivot, then a_i and a_j will be put into different bucket.
  - They will never be compared afterwards.
  - Hence, $X_{i,j} = 0$.

When does a_i and a_j get compared?

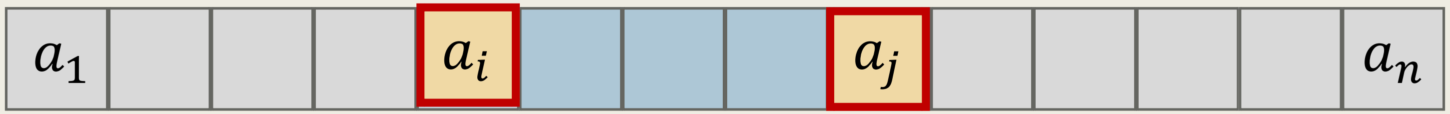


- If one of a_i or a_j is picked as pivot, then they are compared in this iteration.
  - $X_{i,j} = 1$.

When does a_i and a_j get compared?



- If some number smaller than a_i or some number larger than a_j is picked as pivot, then they are put in the same bucket.
  - They *may or may not* be compared in the future.
  - In the next round, the range becomes smaller.



- Let E_t denote the event that some number between a_i and a_j (inclusive) is selected as pivot in the t-th round *for the first time*.

  Then,

$$\Pr[\, X_{i,j} = 1 \mid E_t \,] \;=\; \frac{2}{j - i + 1} \;.$$

- Hence,

$$\Pr[X_{i,j} = 1] \;=\; \sum_{t \ge 1} \Pr[\, X_{i,j} = 1 \,\cap\, E_t \,]$$

$$= \sum_{t \ge 1} \Pr[\, X_{i,j} = 1 \mid E_t \,] \cdot \Pr[\, E_t \,] \;=\; \frac{2}{j - i + 1} \;.$$

Alternative Analysis

- Suppose that the input numbers after sorted are

$$a_1 < a_2 < \cdots < a_n \, .$$

- We obtain

$$E[X] \; = \sum_{1 \le i < j \le n} \Pr\left[X_{i,j} = 1\right] \; \le \; 2nH_n \; = \; O(n \log n)$$

- If the numbers are *not distinct*, the bound *becomes better*.

Problem B

- The number of operations modern CPUs can do in 1 sec is *roughly* 10^9.
- You may want to calculate the allowable time complexity of any algorithm for this problem.
  - Design an algorithm that fulfills the task in time.

Problem C

- You may want to use the cross-products of 2D-vectors to design a valid test for tangent points.