Lecture 11: Randomization in algorithm design and more

Junzhe Zhang

12/17/2023

1 Quicksort

Input: Array A of n items
Goal: sort the given array in place

Strategy:

- If n is small: use brute force.
- Else: pick a pivot in the array, call it element x = A[i]. Then, we perform partition k = Partition(A, i, n, x). Recurse the problem on A[1, k-1] and A[k+1, n]

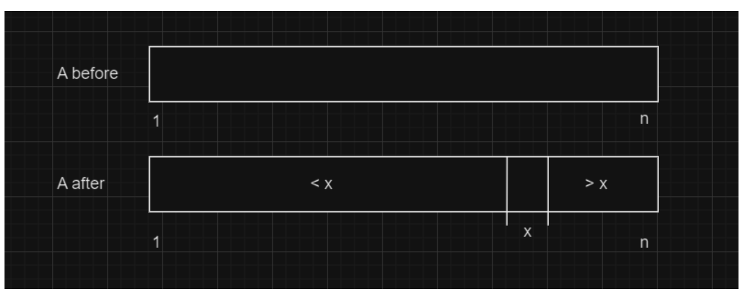

Figure 1: Partition concept

It is well known that the worst case running time is $\theta(n^2)$, which is when x is the smallest or largest. It is shown as $T_{bad}(n) = T_{bad}(n-1) + \Theta(n)$.

Claim: If pivot is chosen randomly and uniformly, with probability $\frac{1}{n}$, the algorithm runs in $\Theta(nlogn)$.
Proof #1: Consider 2 cases:

- case 1: rank of pivot is between $\frac{n}{4}$ and $\frac{3n}{4}$. In this case, sizes of subproblems are always $\leq \frac{3}{4}n$. Running Time $T = T(n_1) + T(n_2) + O(n)$.

Figure 2: Case 1

For the case that gives max imbalance $n_1 = \frac{3n}{4}, n_2 = \frac{n}{4}$, $T \leq T(\frac{3n}{4}) + T(\frac{n}{4}) + O(n)$.

- case 2: $n_1 = n-1, n_2 = 1$. $Probability \leq T(n-1) + O(n) \leq T(n) + O(n)$

$T(n) = expected\ runtime\ of\ input\ sizen$

$$T(n) = \frac{1}{2}[Runtime\ of\ the\ good\ case] + \frac{1}{2}[Runtime\ of\ the\ bad\ case]$$

$$T(n) \leq \frac{1}{2}[T(\frac{3n}{4}) + T(\frac{n}{4}) + O(n)] + \frac{1}{2}[T(n) + O(n)]$$

$$T(n) \leq \frac{1}{2}[T(\frac{3n}{4}) + T(\frac{n}{4}) + T(n)] + O(n)$$

$$2T(n) \leq T(\frac{3n}{4}) + T(\frac{n}{4}) + T(n) + O(n)$$

$$T(n) \leq T(\frac{3n}{4}) + T(\frac{n}{4}) + O(n)$$

$$T(n) = O(nlogn)$$

Proof #2: Count expected number of comparisons made by Quicksort. We focus on 2 elements: X_i, X_j, which are elements with rank i and rank j in A. P_{ij} = probablity that we ever compare X_i and X_j. Watch i, j as Quicksort proceeds.

$$pivot\ rank \begin{cases} < i : nothing \\ = i : 1comparison \\ i < index < j : 0\ comparison,\ never\ compared\ again\ because\ problem\ splitted\ to\ 2\ sides \\ = j : 1comparison \\ > j : nothing \end{cases}$$

$$P_{ij} = \frac{2}{j-i+1} \tag{1}$$

Note that the number of elements is the range i to j including i, j.
Let's use indicator random variable:

$$X_{ij} = I\{X_i\ is\ compare\ to\ X_j\} \begin{cases} 1\ if\ they\ are\ compare \\ 0\ if\ not \end{cases}$$

$\sum_{ij} = total\ number\ of\ comparisons, 1 \le i < j \le n$ Expected number of comparisons=

$$E[\sum_{ij} X_{ij}] = \sum E[X_{ij}] = \sum_{ij} P_{ij} = \sum_{i<j} \frac{2}{j-j+1} = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{2}{j-i+1} \qquad (2)$$

- $i = 1 : \frac{2}{2-1+1} + \frac{2}{3-1+1} + \frac{2}{4-1+1} + ... + \frac{2}{n-1+1}$
- $i = 2 : \frac{2}{3-2+1} + \frac{2}{4-2+1} + ... + \frac{2}{n-2+1}$
- ...
- $i = n - 1$

Simplifying these sums:

- $\frac{2}{1+1} + \frac{2}{2+1} + \frac{2}{3+1} + ... + \frac{2}{n}$
- $\frac{2}{1+1} + \frac{2}{2+1} + \frac{2}{3+1} + ... + \frac{2}{n-1}$
- $\frac{2}{1+1} + \frac{2}{2+1} + ... + \frac{2}{n-2}$
- ...
- $\frac{2}{1+1}$

If the sum of all these sums is S, then

$$S \le 2n[\frac{1}{2} + \frac{1}{3} + \frac{1}{4} + ... + \frac{1}{n}] \le 2nln(n) = O(nln(n)) = O(nlogn) \qquad (3)$$

2 Randomly Constructed BST

Setup: input:A[1...n] Randomly permute A and build a standard BST by repeated inserts.
Question:

- what is the expected cost? $cost = number\ of\ comparisons$ $+\Theta(1)$, worst case: $\Theta(n^2)$
- what is the expected height? worst case: n - 1

Question #1: The root is involved in $n - 1 comparisons$.

How many comps with x for things inserted later? C =size of the left + right subtrees.

Amazing fact: These comparisons are tame comparisons that Quicksort makes, so they expected number of comparisons is the same! In expectation, the total

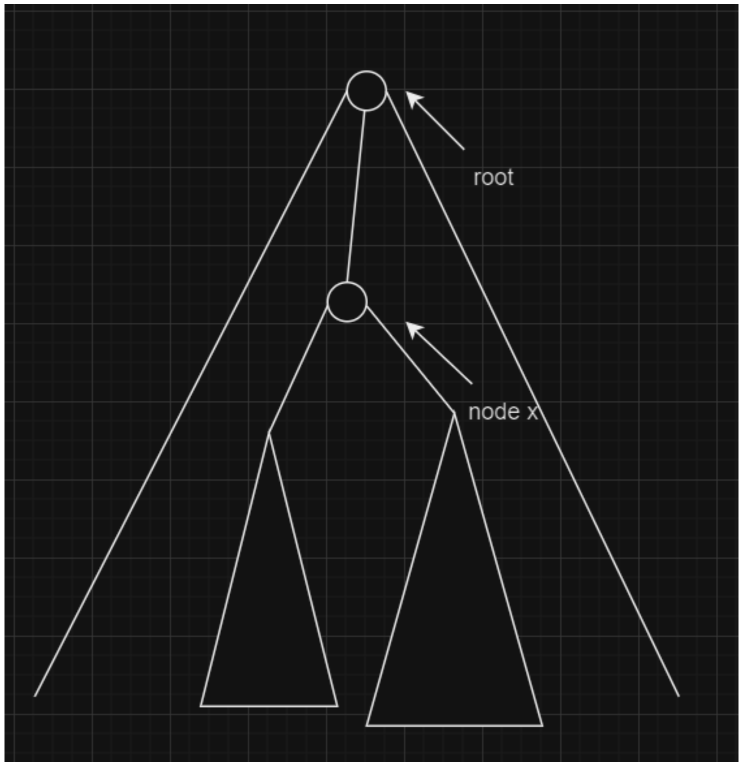

Figure 3: element x in the tree

insertions work is $O(nlogn)$.

Question #2: H(n) = expected height of tree on n elements
H(1) = 0
If root rank is k:

- probability $\frac{1}{2} \rightarrow \frac{n}{4} < k < \frac{3n}{4}$, then $height \leq max(H(\frac{n}{4}, \frac{3n}{4}) + 1$
- probability $\frac{1}{2}$: otherwise $height \leq 1 + H(n-1)$

In expectation,

$$\begin{aligned} H(n) &\leq \frac{1}{2}[height\ of\ the\ good\ case] + \frac{1}{2}[height\ of\ the\ bad\ case] \\ &= \frac{1}{2}[max(H(\frac{3n}{4}), H(\frac{n}{4}))] + \frac{1}{2}[H(n-1)] + 1 \end{aligned} \tag{4}$$

Multiply by 2 on both sides, and subtract H(n):

$$H(n) \leq H(\frac{3n}{4}) + 2 \leq 2log_{\frac{4}{3}}n \approx O(logn) \tag{5}$$

Conclusion: tree is "balanced in expectation".

3 Skip List

Idea: Linked lists are nice because we can easily do insert and delete, but search is a problem.

How it works conceptually: Say we have a sorted list, e.g. 1,5,49,68,103. The skip list can look like this: Search for 11: at:

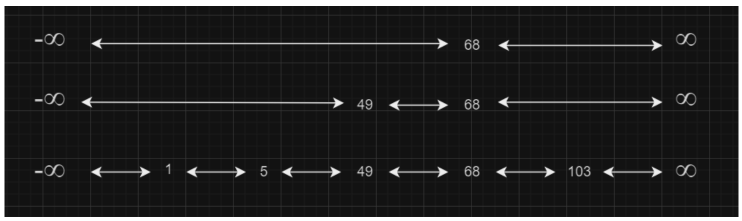

Figure 4: Skip List example

- 1: $-\infty$ to 68, but 68 is too high.
- 2: $-\infty$ to 49, but 49 is too high.
- 3 $-\infty$ to 1, 1 to 5, 5 to 49, 49 too high.
- conclusion: no 11

3.1 Operations

Insert: search and insert where it is supposed to be at the bottom level, then repeatedly flip a coin (prob = p). If success, go up a level and insert again, and so on.
Delete: search and delete from all levels. Without a capacity on number of levels, the worst case insertion time is infinity. Worst-case space is also infinity. Number of levels is infinity.
Cost of Insert(not counting search) = number of levels the element is inserted in.Insert with

- prob = 1 on bottom level (level 0)
- prob = p on level 1
- prob = p^2 on level 2
- ...
- prob = p^i on level i

Repeated trails with probability of stopping is 1-p.
expected number of levels $= 1 \cdot 1 + p \cdot 2 + p^2 \cdot 3 + ...$

$= 1 + p + p^2 + p^3 + ... \quad \rightarrow = \frac{1}{1-p}$
$+p + p^2 + p^3 + ... \quad \rightarrow = \frac{p}{1-p}$
$+p^2 + p^3 + ... \quad \rightarrow = \frac{p^2}{1-p}$

$= \frac{1}{1-p}[1 + p + p^2 + p^3 + ...] = (\frac{1}{1-p})^2 = \frac{1}{(1-p)^2}$

Conclusion:expected cost of insertion is $O(\frac{1}{(1-p)^2}) = 1$ if p is constant.
Expected size: sum of all elements on the number of times they are expected to appear. We know that if p is constant, this is constant O(1) per element.
Total size $= nO(1) = O(n)$

Expected Cost of Delete(not counting search):
If we want to delete a random element, its number of appearance is expected to be $\frac{structure\ size}{n} = \frac{O(n)}{n} = O(1)$, where n is the current number of elements.

Deleting a random element from the structure makes O(1) expected changes.

What if we want to remove an arbitrary (not random) element? Worst case damage = number of levels in this structure.
Search cost: at least = number of levels.

$$Intuition \begin{cases} not\ exactly\ correct. \\ number\ of\ levels\ \approx log_2 n, if\ p = \frac{1}{2} \end{cases}$$

‘ Roughly, structure size $= (n + pn + p^2n + ...) = n(1 + p + p^2 + ...) = \frac{n}{1-p}$

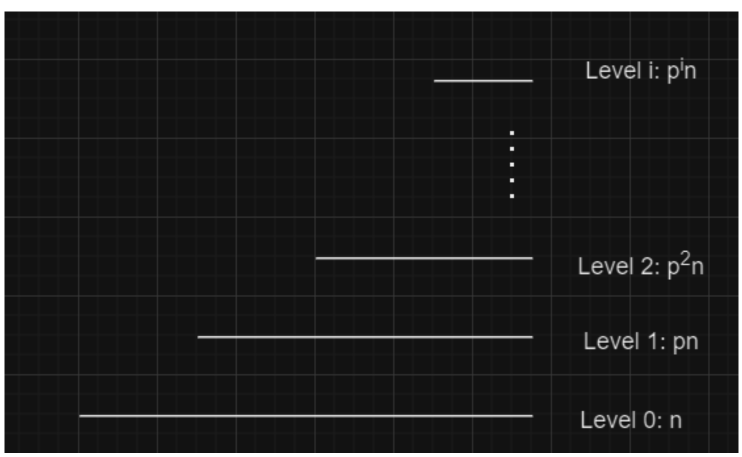

Figure 5: General case

Cost in expectation of level i $\rightarrow$ level i-1
"Gap": consecutive elements in level i-1, but not level i.

- 1 item didn't get promoted to level i with probability (1-p)
- 2 items didn't get promoted to level i with probability $(1-p)^2$
- 3 items didn't get promoted to level i with probability $(1-p)^3$
- ......

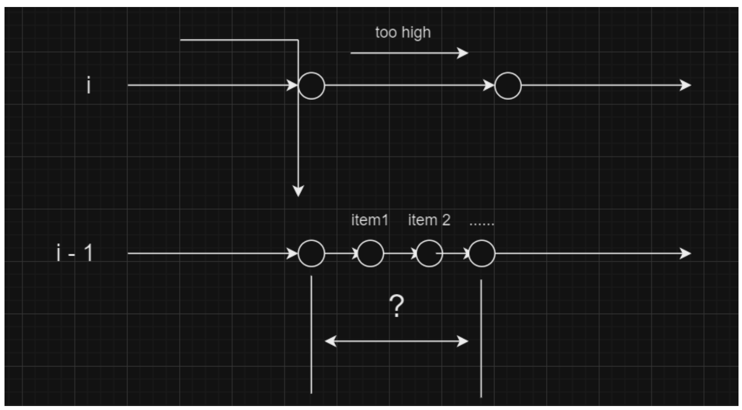

Figure 6: level i and i-1

Expectation:

$$1(1-p)+2(1-p)^2+3(1-p)^3+......=\frac{1}{p^2} \tag{6}$$

Thus, the amount of work walking horizontal per level during a search is O(1) if p is constant.

Final Conclusion: expected cost of search + insert + delete = $O(number\ of\ levels)$

4 Max - 3SAT

Input:3 SAT formula. Example: $(a \vee \overline{b} \vee c) \wedge (\overline{a} \vee b \vee d) \wedge (c \vee d \vee \overline{b})$...
Note: this problem is NP-Hard. Observation: if we pick every variable's truth value randomly and independently (example: $X_i = 0 with probability \frac{1}{2}, X_i = 1 with\ probability\ \frac{1}{2}$), then expected number of true clauses = $\frac{1}{8}m$, where m is the number of clauses.

Proof: prob(a fixed clause is false) = $\frac{1}{8} = \frac{1}{2} \cdot \frac{1}{2} \cdot \frac{1}{2}$.
Find a clause j, let $T_j = I\{Clause\ j\ is\ false\}$, $\sum_{j=1}^{m} T_j$ =number of false clauses.

The expected number of false clauses $= E[\sum_{j=1}^{m} T_j] = \sum_{j=1}^{m} E[T_j] = \sum_{j=1}^{m} prob(clause\ is\ false) = \sum_{j=1}^{m} \frac{1}{8} = \frac{m}{8}$.

3 SAT formula, m clauses and random assignment $\rightarrow \frac{m}{8}$ expected number of false clauses.

Corollary (these are expectations, not guarantees):

1. There is always an assignment with $\geq \frac{m}{8}$ false clauses.
2. OPT $\geq \frac{m}{8}$ for MAX-SAT, $OPT \leq m$
3. A random assignment gives $\frac{m}{8} \geq \frac{OPT}{8}$ false clauses, in expectation.

Approximation factor: a clause is false only when the truth values of all 3 variables are false. So the probability of the clause being true is $\frac{7}{8}$. Therefore, if there are m clauses, the expected number of true clauses should be $\frac{7m}{8}$. But we know the maximum number of true clauses is m, so the approximation factor $\geq \frac{m}{\frac{7m}{8}} = \frac{8}{7}$.

5 Max global cut (NP-Hard)

Undirected Graph G(V,E), global cut: $A, B \subset V, A \cup B = V, A \cap B = \emptyset$.
$Weight\ of\ cut\ (A,B) = w(A,B) = v, u \in E | v \in A, u \in B = number\ of\ edge\ crossings\ in\ the\ cut.$

Problem: given a graph G, find (A,B), we aim to maximize w(A,B).
East approximation: G(V,E) randomly assign every $v \in V$, to A or B with probability $\frac{1}{2}$ independently.
$Prob[uv \in E\ crosses\ the\ cut] = \frac{1}{2}$ Let $X_e = I\{e\ crosses\ the\ cut\}$

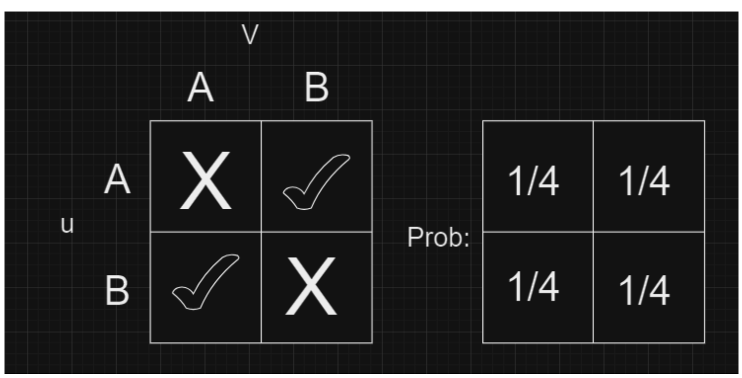

Figure 7: Probability chart

$\sum_{e \in E(G)} X_e = number\ of\ edges\ crossing\ the\ cut\ = E[\sum_e X_e]$

$= \sum_e E[X_e] = \sum_e prob[e\ crosses\ a\ cut] = \sum_e \frac{1}{2} = \frac{m}{2}$, $m = |E(G)|$.
Corollary:

- there always exists a cut of size $\frac{m}{2}$.
- expected size of cut $\geq \frac{m}{2}$.
- $OPT(maxcut) \geq \frac{m}{2}$.
- random choice gives you a 2 approximation.

Again, there is no worst case guarantees or high probability guarantees, and we have no idea what the largest cut is.