CompSci 260P: Week 10

Quiz 2: Solutions

Ryuto Kitagawa

University of California, Irvine

Problem Statement

- Purchase n items for corporate expansion
- Each item costs $\$P$
  - Item i increases in cost by $r_i > 1$ factor each week
- Only buy **one** item each week
- To minimize the total cost of purchases:
  - Sort items in decreasing order of r_i
  - Purchase items in that order
- **Prove this is optimal**

Approach to Problem

- Identify how much buying each item costs
  - $P \cdot r_i^k$ to buy item i on week k
  - $0 \leq k < n$
- Consider what the solution is asking for
  - Order of picking elements
  - Similar to the Algorithmic Pizza problem!
- We will assume this similarity was **not** thought of

Approach to Problem

- Consider the Greedy solution compared to the OPT
  - Greedy *always* picks the item with the lowest r first
  - Let Greedy have picked r_i first and OPT have picked r_j
- OPT picks $r_j, \ldots, r_i, \ldots$
- Let's swap to $r_i, \ldots, r_j, \ldots$
  - How does this change the sum?

Compare Greedy vs Optimal

$$\text{ORG} = P + P \cdot r_i^k + E$$

$$\text{SWP} = P + P \cdot r_j^k + E$$

- Which one is better?
- $P \cdot r_i^k \overset{?}{>} P \cdot r_j^k$
- Recall that $r_i > r_j$, thus left-hand side is *larger*

Is This Proof Enough?

- Not quite enough!
- Logic only holds for swapping the first item!
  - We need to prove this holds for all swaps

Generalized Proof

$$\mathrm{ORG} = P \cdot r_j^{\ell} + P \cdot r_i^{k} + E$$
$$\mathrm{SWP} = P \cdot r_i^{\ell} + P \cdot r_j^{k} + E$$

- Which one is better?
- $P \cdot r_i^{k-\ell} \stackrel{?}{>} P \cdot r_j^{k-\ell}$
- Recall that $r_i > r_j$, thus left-hand side is *larger*
  - $\ell < k$
- Generalizes to all swaps

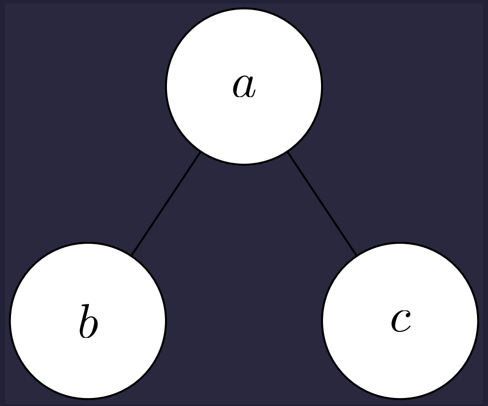
Problem Statement

- We are given a complete binary tree T with $n = 2^d - 1$ vertices
  - Each node has a distinct value
- Find a "local minimum," i.e. a node that is lower than all adjacent nodes
  - Use only $O(\log n)$ inspections of vertex values
- **Crucial Information**: Not necessarily sorted

Approach to Problem

- Start very simple by isolating a part of the problem
  - Consider just the root
  - Checking if it is a local minimum takes $O(1)$ time
- If not, we could recurse on both sides
  - Leads to $T(n) = 2T(n/2) + O(1) = \Theta(n)$
  - **Note:** How do we know that the subproblem is exactly half?
- What if we could recurse to just one side?

Analyze Properties



- Suppose $b < a < c$
- Would c ever be a local minimum?
  - What if we just looked into the left branch?
- $T(n) = T(n/2) + O(1) = \Theta(\log n)$
- But is it correct?

Correctness

- Suppose we go to a node that is less than the current
  - Are we guaranteed to find a local minimum?
- **Yes**
  - Either we find one along the path
  - Or we reach a leaf, which must be a local minimum!