Shuyi Jin sj445
ECE 590: Theory and Practice of Algorithms
November 28, 2024

Homework 7

Question 1. The Index Problem

Description: We can use a modified binary search to solve this problem in O(logn) time. Since the array is sorted in ascending order and contains distinct integers, we can eliminate half of the search space at each step by comparing the value at the middle index with the index itself.

Here is the algorithm and explanation formatted in LaTeX:

—

Algorithm for Finding an Index i Such That $A[i] = i$

```
1) Input: A sorted array $A$ of $n$ distinct integers.
2) Output: An index $i$ (if exists) such that $A[i] = i$, or determine that no such index exists.
3) Steps:
   (a) Initialize low = 1 and high = $n$.
   (b) While low $\leq$ high:
       (i) Compute mid = $\lfloor(\text{low} + \text{high})/2\rfloor$.
       (ii) If $A[\text{mid}] = \text{mid}$, return mid.
       (iii) Else if $A[\text{mid}] > \text{mid}$, set high = mid − 1.
       (iv) Else, set low = mid + 1.
   (c) If the loop ends without finding an index, return "No such index exists".
```

—

The **explanation** of why:

The algorithm uses binary search, which divides the search space in half at every iteration. With an array of size n, the number of iterations required to reduce the search space to one element is $\log_2 n$. Each iteration involves constant-time operations (comparison and index calculation), so the overall time complexity is $O(\log n)$.

Question 2. Party Planning!

(a) Greedy Algorithm Description. The greedy algorithm proceeds as follows:

1) **Traverse the Tree**: Perform a depth-first traversal(DFT,smiliar to DFS) of the tree. Start from the root and process the nodes in a post-order manner (process all children of a node before the node itself).
2) **Decision Rule**: For each node:
   - If none of the node's children are invited, invite the node.
   - Otherwise, do not invite the node.
3) **Mark Decisions**: Maintain a record of whether each node is invited or not.

This ensures that we make a decision for each node based only on its immediate children.

(b) Worst-Case Asymptotic Runtime. The traversal of the tree in a depth-first manner processes each node exactly once. For each node, we check its children to determine whether they are invited or not.

Let n be the number of nodes in the tree. The time complexity is:

$$O(n)$$

This is because the algorithm performs $O(1)$ operations per node during the traversal.

(c) Proof of Correctness Using Induction. We use mathematical induction to prove that the algorithm maximizes the number of invited employees under the constraint that no two adjacent nodes are invited.

Base Case: For a tree with a single node (a root with no children):

- The algorithm invites the root if it has no children, which is the optimal solution.

Inductive Hypothesis: Assume that for any tree of height k or less, the algorithm correctly computes the largest number of employees that can be invited under the constraints.

Inductive Step: Consider a tree of height $k + 1$:

1) During the post-order traversal, when processing a node v:
   - The algorithm already knows the optimal invitation decisions for all of v's children (since their subtrees have height $\leq k$).
2) **Case 1:** If none of v's children are invited, the algorithm invites v, which is valid since it avoids adjacent invitations.
3) **Case 2:** If at least one child of v is invited, the algorithm skips v. This is also valid and ensures that adding v would not create a conflict.

By ensuring decisions are made based only on the children of v, and since the children have already been processed optimally, the decision at v preserves overall optimality.

Conclusion: By induction, the algorithm produces the optimal solution for any tree.