Problem Set 4, Question 2

Input : Tree T

Output : A subset of edges M of T, that are pairwise non-adjacent.

Initially M is empty, $i = 1$

Repeat till T is empty

$e_i = (u_i, v_i)$ such that v_i has the maximum depth in T

$M = M \cup \{e_i\}$

Delete all edges that are incident on u_i from T

Return M.

Running Time

- Depth of vertices can be pre-computed in $O(n)$ time by a tree traversal algorithm

- Since, we are selecting vertex with maximum depth at any point, the tree T remains connected throughout the algorithm

- At any iteration of the loop, an edge is added to the solution and possibly several edges are deleted. Those edges are not further considered by the algorithm.

$\therefore$ Total time spent by the loop

$$= O(\#\text{edges}) = O(n)$$

Total running time $= \underline{\underline{O(n)}}$.

<u>Proof of correctness</u>

Feasibility holds since once an edge is added to solution, adjacent edges are deleted.

Let G be the solution returned by the algorithm. Let $G = \{e_1, e_2, \ldots, e_k\}$ be ordered by non-increasing order of depth. and edges of same depth are ordered from left to right.

Let $A = \{a_1, \ldots, a_\ell\}$ be another maximal feasible solution ordered in the same way as G.

We will show a proof by exchange
Let i be the first index such that $e_i \neq a_i$.

Let $e_i = (u, v)$ such that u is the parent of v.
- G does not contain an edge (v, w) such that v is parent of w [By definition of greedy algorithm)
- A also does not contain such an edge, since $a_j = e_j$, for all $j < i$.

- By maximality of A, there exists $(u, w) \in A$.
Let $A' = \left(A \setminus \{(u, w)\}\right) \cup \{(u, v)\}$.

A' is feasible since no other edge is incident on u or v.

Also, $|A'| = |A|$.

By repeating the above step multiple times, we can convert A into G.

This implies G is optimal.

Problem Set 5 - Qn 5

Subproblem

$OPT[i]$: returns TRUE if strings $A[1 \ldots i]$ and $B[1 \ldots i]$ can be partitioned into words at the same indices.

$OPT[n]$ is the desired answer.

Recurrence

$OPT[0] = TRUE$

$$OPT[i] = \bigvee_{\substack{j \leq i \\ \text{s.t } A[j \ldots i] \text{ and } \\ B[j \ldots i] \text{ are} \\ \text{valid words.}}} OPT[j-1]$$

Proof of correctness of OPT[i] :

By Mathematical Induction on i

Base Case:

$i = 0$, Trivially true.

Induction Step

Induction Hypothesis : OPT[j] is true for $j < i$.

Assume OPT[i] is not correct.

Case 1 : OPT[i] returns TRUE, but the correct answer is FALSE i.e. there does not exist a partition as desired.

Since $OPT[i]$ returns TRUE, there exists $j \leq i$ s.t $A[j \cdots i]$ and $B[j \cdots i]$ are valid words and $OPT[j-1] = TRUE$.
This implies $OPT[j-1]$ is not correct, a contradiction.

<u>Case 2</u>: $OPT[i]$ returns FALSE but the correct answer is TRUE.
i.e., there exists a partition of $A[1 \cdots i]$ and $B[1 \cdots i]$. Let j be the index such that the last word in this partition starts at index j.
$\therefore A[j \cdots i]$ and $B[j \cdots i]$ are valid words.

Since, $OPT[i]$ returned FALSE, none of the terms on the R.H.S returned TRUE.

This implies, $OPT[j-1] = FALSE$, a contradiction

Hence, Proved.

Implementation

The values of OPT[i] are computed in the increasing order of i.

$OPT[0] = 0$
for $i = 1$ to n
$$OPT[i] = \bigvee_{\substack{j < i \\ \text{s.t.} \ldots}} OPT[j-1]$$

Running Time

No. of subproblems $= O(n)$
Time to compute one subproblem $= O(n)$

Since OR is taken over (possibly) all $j-1 < i$.

$\therefore$ Total time $= O(n^2)$.