University of Washington
Department of Computer Science and Engineering
CSE 417, Winter 2020

February 25, 2020

Homework 7 problem 1

Problem 1 (10 points) Weighted Independent Set on a Path:

The weighted independent set problem is: Given an undirected graph $G = (V, E)$ with weights on the vertices, find an independent set of maximum weight. A set of vertices I is independent if there are no edges between vertices in I. This problem is known to be NP-Complete.

For a simpler problem, consider a graph that is a path, where the vertices are $v_1, v_2, \ldots, v_n$, with edges between v_i and v_{i+1}. Suppose that each node v_i has an associated weight w_i. Give an algorithm that takes an n vertex path with weights and returns an independent set of maximum total weight. The run time of the algorithm should be polynomial in n.

Answer:

algorithm:

```python
# G = [v1, v2, v3 ...]
# dp[i] is the maximum total weight of v1 to vi vertices
def maximum_total_weight(G):
    n = len(G)
    dp = [0 for i in range(n+1)]
    dp[1] = G[0]
    for i in range(2,n+1):
        dp[i] = max(dp[i-1], dp[i-2] + G[i-1])
    return dp
```

Correctness proof by induction:
base case:
By defination, when i = 0, the subset of vertices is null, therefore the optimal total weight is zero.

induction hypothesis:
For $i > 0$: dp[i] = max(dp[i-1], dp[i-2] + G[i-1])
(dp[i] as the optimal solution of subproblem of $\{v_1, v_2, \ldots, v_i\}$)

There are only two possibilities considering each vertex:

(1) vertex i is in optimal result:

Then v_{i-1} can not be part of the problem, and the optimal of subproblem v_{i-1} (dp(i-1)) should not be considered as optimal solution component, since v_{i-1} can not link directly to v_i. Then the i - 2, the subproblem before i - 1 should be considered as part of optimal solution with no directly linked vertex, hence dp(i) = dp(i-2) + v_i

(2) vertex i is not in optimal result:

Then the optimal of subproblem v_{n-1} (dp(i-1)) should be considered if without v_i, so simply dp(i) = dp(i-1).

Since the function dp(i) always correctly choose the optimal result with subset $\{v_1, v_2, \ldots, v_i\}$, and dp(i)'s result built opon dp(i-1) and (i-2), hence by choosing one with more maximum weight sum, the function garantees to correctly find the local optimal result. Adding to final vertex v_n which reachs the full problem.

complexity:
Since on each subproblem, the algorithm only traces and compares two past result dp(i-1) and dp(i-2) with a constant time, then there are n subproblem in total, therefore the time complexity is $O(n)$