CSE 102 Spring 2021
Homework Assignment 7

Jaden Liu
University of California at Santa Cruz
Santa Cruz, CA 95064 USA

May 24, 2021

1 HW7

1. **0-1 Knapsack:** (2 points) Why dynamic programming algorithm is not a polynomial algorithm? What is a proper way of describing the computational complexity of the dynamic programming solution to the 0-1 knapsack problem?

Solution. In the dynamic programming algorithm, we need to fill up a table with n*W size. Therefore the time complexity is O(nW). Since the input size = $logw_1 + logv_1 + logw_2 + logv_2 + ... + logw_n + logv_n + logW = nlowW + nlogV$, and V can be accounted for the in the constant term. Thus, the input size = nlogW, which means the runtime is exponential in the input length.[1] □

2. **Greedy-Set Cover:** (3 points) The greedy algorithm is a $\ln n$- approximation to the optimal solution. Show that for any integer n (you may assume that n is a power of 2), there is an instance of set cover problem with the following properties: (i) There are n elements in the base set. (ii) The optimal cover uses just two sets. (iii) The greedy algorithm picks at least log n sets.

Proof. I. Since $n_t \leq n_0(1-(\frac{1}{k})^t) < ne^{-t/k}$, n_t is strictly less than $ne^{-lnn} = 1$ at $t = klnn$, which means no elements remain to be covered. Thus there are n elements in the base case.
II. Just purposedly give a set that only needs two set covering the whole set.
III. Since we have proved that $n_t \leq n_0(1-(\frac{1}{k})^t) < ne^{-t/k}$, then the ratio between the optimal solution and greedy algorithm will never be greater than log n sets. □

3. **Greedy-Set-Cover:** (3 points) CLRS, 35.3.1, Page 1122. Please note that the letters are vertices and words are sets. The problem is to find the greedy solution, an approximation, to the least number of vertices (letters) that cover all the sets (words). [arid, dash, drain, heard, lost, nose, shun, slate, snare, thread]

Proof. First we find thread(6 letters), then shun(4 letters), lost(2 letters) and drain(1 letters). □

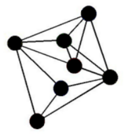
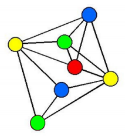
4. **Graph Coloring:** (4 points)

   (a) (2 points) A graph G is 2-colorable iff it is bi-partite. [A graph is bi-partite if its vertices can be partitioned into two sets of vertices X and Y so that every edge has one end in X and the other end in Y.]

   (b) (2 points) Provide an example of a planar graph (with ≤ 9 vertices) that is not 3-colorable, that is the graph requires at least 4 colors, that is no adjacent nodes have same color. Prove that 4 colors are required.

Proof. a):Let G be a 2-colorable graph, which means we can color every vertex either red or blue, and no edge will have both endpoints colored the same color. Let A denote the subset of vertices colored red, and let B denote the subset of vertices colored blue. Since all vertices of A are red, there are no edges within A, and similarly for B. This implies that every edge has one endpoint in A and the other in B, which means G is bipartite.[3]

Conversely, suppose G is bipartite, that is, we can partition the vertices into two subsets V1, V2 every edge has one endpoint in V1 and the other in V2. Then coloring every vertex of V1 red and every vertex of V2 blue yields a valid coloring, so G is 2-colorable. □

Proof. b):  $\Rightarrow$  [2]

In this graph, we can consider the red dot as the center dot. Consider the four point on the left top. They connected each other, so we needs at least 4 colors to represent the graph. □

5. **Greedy-Clustering:** (4 points) Greedy algorithm: pick any point as first cluster center, pick the next one farthest from the previously selected ones; thus choosing k centers and then assigning remaining points to the closest center. Prove that the above greedy algorithm provides a 2-approximation to the optimal solution. Details available

Proof. Let $x \in X$ be the point farthest from $u_1, ..., u_k$ (in other words the next center we would have chosen, if we wanted $k+1$ of them), and let r be its distance to its closest center. Then every point in X must be within distance r of its cluster center. By the triangle inequality, this means that every cluster has diameter at most $2r$ □

35-1 Bin packing
Suppose that we are given a set of n objects, where the size s_i of the ith object satisfies $0 < s_i < 1$. We wish to pack all the objects into the minimum number of unit-size bins. Each bin can hold any subset of the objects whose total size does not exceed 1.

a. Prove that the problem of determining the minimum number of bins required is NP-hard. (*Hint:* Reduce from the subset-sum problem.)

The ***first-fit*** heuristic takes each object in turn and places it into the first bin that can accommodate it. Let $S = \sum_{i=1}^{n} s_i$.

b. Argue that the optimal number of bins required is at least $\lceil S \rceil$.

c. Argue that the first-fit heuristic leaves at most one bin less than half full.

d. Prove that the number of bins used by the first-fit heuristic is never more than $\lceil 2S \rceil$.

e. Prove an approximation ratio of 2 for the first-fit heuristic.

f. Give an efficient implementation of the first-fit heuristic, and analyze its running time.

Proof. b): Assume the we can add partial of the element, then the total bin we need is $\lceil S/1 \rceil$. Hence the optimal is at least $\lceil S \rceil$.
c): If there is more than 1 bin that is half full, then we can combine these bin together. Thus, generating a contradiction, which proves that at most 1 half full bin.
d,e): $OPT \geq \sum_{i=1}^{n} a_i > \frac{m-1}{2}$, thus $2OPT > m$
f): Given a object in the list, find if it can be put into the present bin, otherwise put into the next bin(or create a new bin), repeat this process until the end of object list. The running time is $O(n^2)$ in the worst case. □

References

[1] Mark Gritter, https://www.quora.com/What-is-a-proper-explanation-of-how-the-dynamic-programming-knapsack-problem-is-not-polynomial-but-pseudo-polynomial

[2] https://stepik.org/lesson/390640/step/3

[3] https://courses.cs.duke.edu//spring19/compsci230/Notes/lecture15.pdf