

Homework Ten, for Mon 12/5

CSE 101

1. The *cut property* makes it possible to construct minimum spanning trees greedily, for instance by using Kruskal's algorithm. In this problem, we will examine a different property that also enables a greedy solution. Here it is:

Pick any cycle in the graph, and let e be the heaviest edge in that cycle. Then there is a minimum spanning tree that does not contain e .

- (a) Prove this *cycle property* carefully.
- (b) Use the property to justify the following MST algorithm. The input is an undirected graph $G = (V, E)$ with edge weights $\{w_e\}$.

```
sort the edges according to their weights
for each edge  $e \in E$ , in decreasing order of  $w_e$ :
    if  $e$  is part of a cycle of  $G$ :
         $G = G - e$  (that is, remove  $e$  from  $G$ )
return  $G$ 
```

- (c) On each iteration, the algorithm must check whether there is a cycle containing a specific edge e . Give a linear-time algorithm for this task, and justify its correctness.
 - (d) What is the overall time taken by this algorithm, in terms of $|E|$? Explain your answer.
2. In many of the problems we have studied, there can be several optimal solutions: several alternative shortest paths, several alternative ways to optimally schedule jobs, and so on. The same holds for minimum spanning trees. However, show that if all the edge weights are distinct, then there is a *unique* MST.
 3. You have a graph $G = (V, E)$ with edge weights $w(\cdot)$, and someone has already given you a minimum spanning tree $T = (V, E')$ of this graph. Suppose, however, that you now need to *increase* the weight of one particular edge e . Does the MST change? If so, show how to compute the new MST in just linear time. You should consider two cases: when $e \notin E'$ and when $e \in E'$.
 4. *One-dimensional set cover*. Suppose we are given points x_1, \dots, x_n on the real line (that is, each x_i is a real number). We are also given a collection of intervals $[s_1, e_1], \dots, [s_m, e_m]$. Show that there is an efficient greedy algorithm that finds the minimum number of intervals needed to cover all the x_i .
 5. In class, we studied Horn formulas: a special class of Boolean formulas for which satisfiability can be determined in linear time. There is a different class of Boolean formulas that can also be solved efficiently, called *2-CNF* formulas. In fact, the way to handle these is to convert them into graphs and find strongly connected components! Do problem 3.28 in the book.

6. Problem 6.1. For this problem, use the following subproblem definition: for each $1 \leq j \leq n$,

$$S(j) = \text{maximum sum of any contiguous subsequence ending exactly at position } j.$$

Note that a subsequence can have length zero, in which case the sum is also zero. Thus no $S(j)$ should ever be negative. The final answer we want is $\max_j S(j)$.

7. Problem 6.3.

- (a) Solve the problem as stated in the book. Use the following subproblem: for all $1 \leq j \leq n$,

$$P(j) = \text{maximum profit achievable using only the first } j \text{ locations.}$$

- (b) Once you have solved the problem by dynamic programming, consider this greedy approach:

Repeat:

Pick the location i with highest profit p_i

Remove all locations within k miles of location i

Do you think this will work? Why or why not?

8. Recall the *string reconstruction* problem from class: You are given a string of n characters $x[1 \dots n]$, which you believe to be a corrupted text document in which all punctuation has vanished (so that it looks something like “itwasthebestoftimes...”). You wish to reconstruct the document using a dictionary, which is available in the form of a Boolean function $\text{dict}(\cdot)$: for any string w ,

$$\text{dict}(w) = \begin{cases} \text{true} & \text{if } w \text{ is a valid word} \\ \text{false} & \text{otherwise.} \end{cases}$$

In class, we found a dynamic programming algorithm that determines whether $x[\cdot]$ is a sequence of valid words, and runs in $O(n^2)$ time, assuming calls to dict take unit time. We’ll now consider an alternative, graph-theoretic approach.

- (a) Consider a directed graph $G = (V, E)$ in which $V = \{1, 2, \dots, n + 1\}$, and there is an edge (i, j) whenever $i < j$ and $x[i \dots j - 1]$ is a valid word. Show that $x[\cdot]$ is a valid sequence of words if and only if node $n + 1$ is reachable from node 1 in G . How long does this graph-based algorithm take?
- (b) Often there are several alternative ways to reconstruct a string. For instance, “potato” could be reconstructed as a single word, or as a sequence of three words, “pot”, “a”, “to”. In such situations, we would like a *minimal* reconstruction: one involving the minimum number of words. Show how to solve this problem using a graph algorithm; it should output the actual minimal sequence of words. What is the running time?

9. Problem 6.7. For this one, a natural subproblem is

$$T(i, j) = \text{length of longest palindromic subsequence of } x[i \dots j].$$

10. Problem 6.21.

11. Problem 6.22. A good subproblem to use is the following: for any $j \leq n$ and $s \leq t$,

$$T(j, s) = \text{true if some subset of } a_1, \dots, a_j \text{ adds up to } s.$$