Midterm 2

- **The exam has 4 questions, is worth 100 points, and will last 120 minutes.**
- **Each question is worth 25 points.** Not all parts of a problem are weighted equally.
- Read the instructions and the questions carefully first.
- Begin each problem on a new page.
- Be precise and concise.
- The problems may **not** necessarily follow the order of increasing difficulty.
- If you use any algorithm from the textbook as a black box, you can rely on the correctness of the quoted algorithm. If you modify an algorithm from the textbook, you must explain the modifications precisely and clearly, and if asked for a proof of correctness, give one from scratch or give a modified version of the textbook proof of correctness.
- Good luck!

1 Smorgasbord Short Answer

(a) Consider a Huffman code for the character-frequency pairs $(A, \frac{1}{2}), (B, \frac{1}{8}), (C, \frac{1}{8}), (D, \frac{1}{8}), (E, \frac{1}{16}), (F, \frac{1}{16})$. Draw a Huffman tree (mid-exam clarification: removed "break ties alphabetically").

(b) Show that the following Horn formula is not satisfiable:

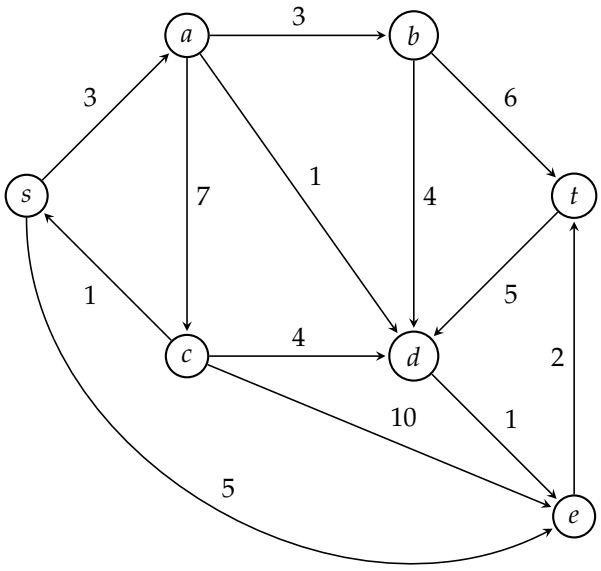
$$\Rightarrow w, \Rightarrow x, (v \wedge w \wedge x) \Rightarrow y, (w \wedge x) \Rightarrow z, (w \wedge x \wedge y) \Rightarrow z, (\bar{y}), (\bar{v} \vee \bar{x}), (\bar{w} \vee \bar{x} \vee \bar{z}).$$

(c) Suppose G has an MST T with edge weights 1, 4, 4, and 8. Argue why G cannot have another MST T' with edge weights 2, 3, 6, and 6.

(d) Draw a graph of four nodes A, B, C, D with **unique** edge weights such that it has a cycle whose minimum edge is *not* part of the MST of the graph.

(e) Consider the knapsack problem with repetition (as discussed in lecture) with a weight bound of W. Suppose the weight bound is increased to W^2. Explain why this increases the runtime of the DP algorithm by an exponential factor.

(f) Suppose you are given the graph G as below. Argue that the maximum $s - t$ flow in this graph is 5.



(a) One answer is: $A = 0, B = 100, C = 110, D = 111, E = 1010, F = 1011$. See Figure 1.

(b) From the first two singleton implications ($\Rightarrow w$ and $\Rightarrow x$), we see that w and x must both be True. Then the implication $(w \wedge x) \Rightarrow z$ forces z to be True as well. But this contradicts the last clause, $(\bar{w} \vee \bar{x} \vee \bar{z})$, so there is no satisfying assignment.

(c) Suppose for the sake of contradiction G had an MST T' with edge weights 2, 3, 6, and 6. Consider the edges of T' with the weight 1 edge – there must be a cycle in this set of edges (since T' is a tree, so adding any edge results in some cycle). But then removing the maximum weight edge in this cycle results in a spanning tree with weight less than 17 (since the edge we remove has weight > 1), which is a contradiction. Hence T' cannot exist.

(d) Consider the example in Figure 2.

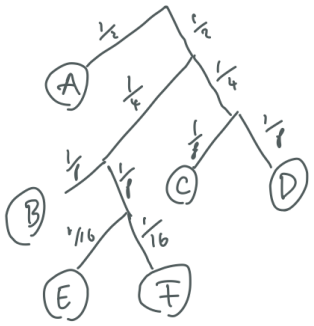


Figure 1: Sample Q1a Solution

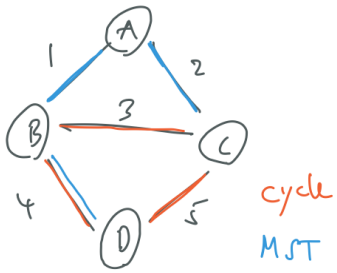


Figure 2: Sample Q1d Solution

(e) The DP algorithm for the knapsack problem discussed in lecture runs in $\Theta(nW)$ time, where n is the number of elements and W is the weight bound. Since the total number of bits m needed to represent the problem instance is $O(n \log W)$, this is exponential time; in particular, it could be that $W = 2^n$, and the time expressed in m alone would be $\Theta(\sqrt{m}2^{\sqrt{m}})$. So, replacing W with W^2 would give us a new runtime of $\Theta(\sqrt{m}2^m)$, which is an exponential factor ($\Theta(2^{\sqrt{m}})$) larger.

(f) By applying the max-flow min-cut theorem to the cut $\{b, t\}$, we see that the $s - t$ flow is at most 5. Moreover, a flow of 5 is achievable by sending 3 units of flow along $s \to a \to b \to t$ and 2 units along $s \to e \to t$.

2 Fruits

Jack Sparrow is shopping for apples, bananas, and cantaloupes at Raider Moe's, in preparation for his upcoming voyage. Jack doesn't want to get scurvy, so he needs to buy fruit containing at least 4500mg of vitamin C in total. Moreover, Jack's ship can carry at most 100 pounds of fruit. Each fruit has different properties, given in the table below:

Fruit	Weight per fruit	Vitamin C per fruit	Calories per fruit	Cost per fruit
Apple	0.4 pounds	8 mg	100 calories	4 copper pieces
Banana	0.3 pounds	10 mg	100 calories	5 copper pieces
Cantaloupe	2 pounds	100 mg	300 calories	50 copper pieces

Jack wants to maximize the total number of calories in his purchased fruit.

Note: You may assume that Jack can purchase fractional quantities of fruit.

(a) Write down (but do not solve) a linear program that Jack can solve to obtain the optimal quantities of apples/bananas/cantaloupes he should buy. (Ignore the cost per fruit for this part.)

(b) Argue that the optimal number of apples Jack buys is 0. (Again, ignore the cost per fruit in this part.)

(c) What is the minimum number of copper pieces Jack needs to spend on fruit to meet the vitamin C and weight constraints? Show that Jack cannot spend less than your minimum and show that the minimum is achievable. (Ignore the calories per fruit in this part.)

Hint: Compare the cost (in copper pieces) with the amount of vitamin C for each fruit.

(a) Jack can solve the following LP:

$$\begin{aligned} \max \quad & 100a + 100b + 300c \\ \text{subject to:} \quad & 0.4a + 0.3b + 2c \leq 100 \\ & 8a + 10b + 100c \geq 4500 \\ & a, b, c \geq 0. \end{aligned}$$

Here, a, b, and c represent the number of apples, bananas, and cantaloupes Jack buys, respectively.

(b) The intuition is that Jack can swap out any apples he buys for bananas since apples are "worse" than bananas in all aspects of the problem. Formally, let (a^*, b^*, c^*) be the optimal number of apples, bananas, and cantaloupes Jack buys. Suppose for the sake of contradiction that $a^* > 0$. Then $(0, b^* + \frac{4}{3}a^*, c^*)$ increases the number of calories:

$$100\left(b^* + \frac{4}{3}a^*\right) + 300c^* = \frac{400}{3}a^* + 100b^* + 300c^* > 100a^* + 100b^* + 300c^*$$

while keeping the weight the same and increasing the total Vitamin C:

$$0.3\left(b^* + \frac{4}{3}a^*\right) + 2c^* = 0.4a^* + 0.3b^* + 2c^*$$

$$10\left(b^* + \frac{4}{3}a^*\right) + 100c^* = \frac{40}{3}a^* + 8b^* + 100c^* > 8a^* + 10b^* + 100c^*.$$

Since $(0, b^* + \frac{4}{3}a^*, c^*)$ satisfies the constraints and increases the number of calories, (a^*, b^*, c^*) is not optimal. This is a contradiction, i.e. $a^* = 0$.

(c) Dividing the "Vitamin C" constraint by 2, we have $4a + 5b + 50c \geq 2250$, i.e. Jack must spend at least 2250 copper pieces on fruit. Moreover, only spending 2250 copper pieces is achievable, e.g. by buying 45 cantaloupes. Therefore, 2250 is the minimum number of copper pieces Jack needs to spend on fruit.

3 Wells

There are n cities numbered $1, 2, \ldots, n$, each of which needs water. To obtain water, a city must either drill its own well or be connected via a sequence of pipes to a city which has its own well. It costs w_i for city i to drill its own well and costs $p_{ij} = p_{ji}$ to construct a pipe between city i and city j. A set of drilling locations and pipe constructions is a *viable plan* if all cities have access to water after all drilling/piping is complete.

- *For full credit:* Design an algorithm to find the minimum-cost viable plan.
- *For half credit:* Design an algorithm to find the minimum-cost viable plan **which involves only one well drilling**.

Clearly indicate whether you are giving a full credit algorithm or half credit algorithm. If you attempt both without any indication, we will only grade the first attempt shown on Gradescope. For the option you choose:

(a) Describe your algorithm.

(b) Prove that your algorithm is correct.

(c) Provide a runtime analysis for your algorithm (in terms of n).

Note: Credit for the proof of correctness and the runtime analysis will only be awarded for correct algorithms that run in asymptotically optimal time.

For full credit:

(a) Consider a complete undirected weighted graph $G = (V, E)$ where $V = \{C_1, C_2, \ldots, C_n, W\}$ and each edge $e \in E$ weighted as follows:

$$w_e = \begin{cases} p_{ij} & \text{if } e = (C_i, C_j) \\ w_i & \text{if } e = (C_i, W) \end{cases}.$$

Then compute a minimum spanning tree T of G using either Prim's or Kruskal's and return the plan:

- Drill a well at city i if $(W, C_i) \in T$.
- Construct a pipe between cities i and j if $(C_i, C_j) \in T$.

(b) In a viable plan, all cities must be connected to water through some path. In the graph G above, this is equivalent to "all cities and W must be connected to each other through some path." So, viable plans correspond to connected spanning subgraphs of G. Since all weights in the graph are "costs" and therefore nonnegative, the minimum cost viable plan is the minimum weight spanning subgraph of G, which is the minimum spanning tree. Since Kruskal's or Prim's will correctly find this tree, our algorithm will correctly find the minimum cost plan.

(c) We create a graph with $n + 1$ nodes and $\binom{n+1}{2} = O(n^2)$ edges. At most we will have to construct this graph in its entirety, taking $O(n^2)$ time. Kruskal's and Prim's, then, both take $O(n^2 \log n)$ time to run on this graph; this is $O(n^2 \log n)$ time total.

For half credit:

(a) Consider a complete undirected weighted graph $G = (V, E)$ where $V = \{C_1, C_2, \ldots, C_n\}$ and each edge $e = (C_i, C_j) \in E$ has weight p_{ij}. Compute a minimum spanning tree T of G using either Prim's or Kruskal's and compute the city k with the minimum drilling cost. Then return the plan:

- Drill the well at city k.
- Construct a pipe between cities i and j if $(C_i, C_j) \in T$.

(b) The minimum-cost viable plan which involves exactly one well drilling must connect all cities together with pipes, as all cities must be connected somehow to the one city with the well. Therefore, by the same argument as the full-credit solution, these pipes form the minimum spanning tree in the graph G and so we find them correctly. Separately, we need to drill exactly one well, with no constraints on where it goes, so we must choose the lowest-cost well.

(c) There are n possible wells, so choosing the lowest-cost one takes $O(n)$ time. By the same argument as in the full-credit solution (except with a graph of n nodes), our algorithm takes $O(n^2 \log n)$ time total.

4 Mittens

You are running Toasty Digits, a company that produces mittens. To make sure that your company can meet demands, you are planning out the production for the coming n months. The following information is given to you:

- **Demand:** In month i, the demand will be $d_i \geq 0$ (i.e. you sell exactly d_i pairs of mittens in month i). The total demand for all n months is given by $D = \sum_{i=1}^{n} d_i$.
- **Production:** Your full-time knitting staff can produce at most m pairs of mittens per month. You can hire additional knitters who will produce as many additional mittens as you need, but at a cost of c dollars per pair (whereas you do not pay anything per pair for your full-time staff).
- **Storage:** If, at the end of a month, you have any leftover mittens, you have to store them at a cost. In particular, if you have k pairs of mittens left, you pay $h_k \geq 0$ dollars. You can store at most D pairs of mittens.

Provide a dynamic programming algorithm for computing the minimum total cost required to meet all demand. Your solution should include the following:

(a) A description of your subproblems and how you will use them to compute the final answer.
Hint: Let one of the subproblem parameters be the number of leftover mittens.

(b) A recurrence relation for your subproblems and the relevant base cases.

(c) A justification for your recurrence relation and your base cases.

(d) The order in which to solve the subproblems.

(e) The runtime of solving all subproblems and computing the final answer.

(a) We will define the subproblem $f(i, r)$ as the minimum cost to get to the end of month i, having satisfied all demand so far, with r pairs of mittens left over. The final answer will be $f(n, 0)$: the subproblem corresponding to month n with 0 mittens left.

(b) The recurrence is given by

$$f(i, r) = \min_{0 \leq p \leq r + d_i} F(i, r, p)$$

where $F(i, r, p)$ is the cost incurred during month i if we start it with p pairs of mittens and end it with r pairs left:

$$F(i, r, p) = \begin{cases} h_p + f(i-1, p) & \text{if } m \geq r + d_i - p \\ h_p + c(r + d_i - p - m) + f(i-1, p) & \end{cases}$$

The base cases are $f(0, \cdot) = \infty$, except for $f(0, 0) = 0$

(c) The base cases follow because at the beginning of month 1, we have zero mittens. This makes $f(0, 0)$ the actual starting point and all other $f(0, \cdot)$ impossible.

The recurrence follows because we want to minimize the cost given the number of mittens we have left over after each month. The cost for each month depends on the demand and the number of mittens we have left over and the number we want to have left over. Furthermore, if we have p mittens left over from the month before, we always pay h_p. Furthermore, we need to consider whatever we have paid so far. Finally, if our full-time staff cannot knit enough mittens to cover demand and the number of mittens we want to have left over, even using the previous stock, we must hire outside staff at cost c per pair.

(d) The order of the subproblems need not depend on the number of mittens left, but we need to solve in terms of increasing months, i.e. $f(i-1, r)$ before $f(i, m)$ for all r, m.

(e) We have $O(nD)$ subproblems and each subproblem takes $O(D)$ time to compute. Hence, the runtime is $O(nD^2)$.