# CSE 551: Quiz 8 Solutions

Jamison Weber

November 5, 2021

## Problem 1

Consider the MAX-SAT problem, where we are given a boolean formula $\Phi$ in conjunctive normal form (i.e. a conjunction of disjoint literals), and we seek a satisfying assignment of variables that maximizes the number of clauses satisfied in $\Phi$. Suppose we have invented an approximation algorithm that finds a suboptimal solution to the MAX-SAT problem and I have somehow proven that this algorithm is a $\rho$-approximation of the optimal solution. What can one then conclude about the value of $\rho$?

- [✓ $0 < \rho < 1$]
- $\rho = 1$
- $\rho > 1$
- Nothing meaningful can be concluded about the value of $\rho$ from the information provided.

### Rationale

The key observation here is that the MAX-SAT problem is a maximization problem, therefore any value of $\rho$ would need to be strictly between 0 and 1. If $\rho > 1$, then one is essentially claiming the algorithm finds a solution better than optimal, which is a contradiction. If $\rho = 1$, one is essentially saying the algorithm is optimal, and is therefore not an approximation algorithm.

## Problem 2

Consider an arbitrary, connected, weighted graph. The professor presents this graph to her students and tasks them with finding whether there exists a hamiltonian cycle of a certain total weight. To which complexity class does this task belong?

- It is strictly NP-complete since it is a decision problem to which all problems in NP can be reduced and the problem is in NP.
- It is strictly NP-Hard since it is a combinatorial optimization problem. for which no polynomial time algorithm is currently known.
- It is in P since the problem can be solved in polynomial time.
- It is in NP but not NP-complete since there exists a polynomial time certifier for any instance of the problem.

### Rationale

The class NP is entirely made of decision problems, which this problem formulation conforms to. Also, the decision version of the traveling salesman problem has been shown to be NP-Complete, so statement 1 is correct. Statement 2 is false. If the professor asked for the lowest weight hamiltonian cycle, then this problem would be strictly NP-Hard since it is not known whether a polynomial time certifier exists. To determine whether a certificate solution is correct, one would have to compare it to all other hamiltonian path solutions. A hamiltonian cycle solution can be represented as a permutation of nodes, thus this would require $n!$ comparisons in the worst case. Statement 3 is false, since no polynomial time algorithm is known to be capable of solving this problem. Statement 4 is false, as it has been demonstrated above that the problem is NP-complete.

## Problem 3

Recall the proof of correctness for the 2-approximation algorithm that solves the traveling salesman problem seen in lecture. Which of the following statements justifies the claim $w(TSP) \leq 2w(MST)$? Assume all positive edge weights.

- The tour found by the approximation algorithm is a valid hamiltonian tour. Since $w\,(TSP)$ is the weight of the optimal hamiltonian tour, it cannot be any greater than the weight of the tour found by our algorithm. Finally, the cost of the tour found by our algorithm has twice the weight of a minimum spanning tree.
- Given n nodes, the optimal hamiltonian tour has $n$ edges, and the tour our algorithm gives will have $2(n-1)$ edges. In addition, $n \leq 2\,(n-1)$ for all $n \geq 2$ and all edge weights are positive by assumption.
- Clearly, doubling the weight of any spanning tree will always be at least the weight of any spanning cycle.
- None of the other responses is correct.

### Rationale

Statement 1 is correct. The tour found by our algorithm is a candidate solution, and by definition, the optimum TSP solution cannot have a greater weight than the one found be our algorithm. Statement 2 is not exactly correct. An optimal hamiltonian tour is the lowest weight hamiltonian tour possible in the graph. Removing an edge of this tour forms a spanning tree, though not necessarily a minimum spanning tree. Therefore, we cannot directly compare the number of edges in this way. Statement 3 is false, since one could select a very high weight spanning cycle that could have an even higher weight than twice the weight of a minimum spanning tree.

## Problem 4

Regarding the dynamic programming framework given for the knapsack problem, which of the following statements justifies $V^* \leq nv_{max}$?

- The largest possible total value a subset can have is found when the algorithm considers a subset containing all possible items, and all items are associated with the same value. Any other subset considered would have a total value less than this.
- We enforce this condition to ensure the algorithm runs in polynomial time.
- This represents the total value of any subset containing the item with maximum value.

- We enforce this condition to ensure the algorithm runs in pseudo-polynomial time.

### Rationale

Statement 1 is true and correctly justifies the claim. Statements 2 and 4 are false, since the claim follows naturally and we do not need to enforce the claim to be true by limiting instances we may consider. Statement 3 is clearly false subsets containing the item with maximum value can contain other items that do not have the maximum value.

## Problem 5

Consider the following instance of the knapsack problem in the table below:

| Item | Value | Weight |
|---|---|---|
| 1 | 155,053 | 5 |
| 2 | 851,200 | 8 |
| 3 | 21,939,416 | 10 |
| 4 | 27,326,369 | 12 |

Use the technique shown in lecture to derive the vector:

$$\begin{bmatrix} \hat{v}_1 \\ \hat{v}_2 \\ \hat{v}_3 \\ \hat{v}_4 \end{bmatrix} \tag{1}$$

Assume $\epsilon = 0.015$.

### Correct Response

$$\begin{bmatrix} 2 \\ 9 \\ 215 \\ 267 \end{bmatrix} \tag{2}$$

### Rationale

Recall that for an item $i$ we calculate $\hat{v}_i = \lceil \frac{v_i}{\theta} \rceil = \lceil \frac{v_i n}{v_{max} \epsilon} \rceil$. We now calculate each entry of the vector:

$$v_1 = \lceil \frac{155053 \cdot 4}{27326369 \cdot 0.015} \rceil = \lceil 1.5131 \rceil = 2$$

$$v_2 = \lceil \frac{851200 \cdot 4}{27326369 \cdot 0.015} \rceil = \lceil 8.3065 \rceil = 8$$
$$v_3 = \lceil \frac{21939416 \cdot 4}{27326369 \cdot 0.015} \rceil = \lceil 214.0976 \rceil = 215$$
$$v_4 = \lceil \frac{27326369 \cdot 4}{27326369 \cdot 0.015} \rceil = \lceil 266.6667 \rceil = 267$$

## Problem 6

Regarding the proof of correctness for the knapsack PTAS seen in lecture, which of the following statements justifies the claim

$$\sum_{i \in S} v_i + n\theta \leq (1+\epsilon) \sum_{i \in S} v_i?$$

- We defined $\theta = \frac{v_{max}\epsilon}{n}$ and there cannot be any instances where $v_{max}$ is more than the sum of all values in the subset chosen by our algorithm.
- We defined $\theta = \frac{v_{max}\epsilon}{n}$ and there cannot be any instances where $v_{max}$ is less than the sum of all values in the subset chosen by our algorithm.
- We defined $\theta = \frac{v_{max}\epsilon}{n}$ and $v_{max}$ may or may not be present in the subset chosen by our algorithm.
- We defined $\theta = \frac{v_{max}\epsilon}{n}$ and there cannot be any instances where $v_{max}$ is more than the sum of all rounded values in the subset chosen by our algorithm.

### Rationale

The first statement is true. Recall that we assumed every item was capable of fitting in the knapsack on its own. Now suppose $v_{max} > \sum_{i \in S} v_i$. This is a contradiction since the dynamic programming algorithm is optimal on the rounded instance, and a better solution would just be to have the item corresponding to $v_{max}$ be the only item in the knapsack, which we assume can fit. This allows us to make the following inference:

$$\sum_{i \in S} v_i + n\theta = \sum_{i \in S} v_i + v_{max}\epsilon \leq \sum_{i \in S} v_i + \epsilon \sum_{i \in S} v_i = (1+\epsilon) \sum_{i \in S} v_i$$

Statement 2 is simply false for the reasons described above. Statement 3 is true but not relevant to the claim. Statement 4 is false, since it is even likely that $v_{max}$ is greater than the sum of all rounded values chosen by the algorithm (See problem 5).

## Problem 7

[Probability theory review] Suppose we have $n$ containers with open lids. For each of these containers we decide to randomly close each lid with probability $\frac{3}{4}$. Let random variable $X_i$ be equal to 1 if the lid of bin $i$ becomes closed and zero otherwise. Then let $X = \sum_{i=1}^{n} X_i$. What is the value of $E[X]$? What does this value represent?

- $\frac{3n}{4}$, the expected number of bins with closed lids.
- $\frac{3n}{4}$, the expected number of bins with open lids.
- $\frac{n}{4}$, the expected number of bins with closed lids.
- $\frac{n}{4}$, the expected number of bins with open lids.
- None of the other responses is correct.

### Rationale

We begin by calculating $E[X_i]$. This is given by $Pr[X_i = 1] \cdot 1 + Pr[X_i = 0] \cdot 0 = Pr[X_i = 1] = \frac{3}{4}$. We can easily calculate $E[X]$ using the calculation above and the rule of linearity of expectation.

$$E[X] = E[\sum_{i=1}^{n} X_i] = \sum_{i=1}^{n} E[X_i] = \sum_{i=1}^{n} Pr[X_i = 1] = \sum_{i=1}^{n} \frac{3}{4} = \frac{3n}{4}$$

Since we defined each $X_i$ to be 1 if bin $i$ is closed, $E[X]$ represents the total number of bins with closed lids.

## Problem 8

In lecture, we saw as a preliminary a 2-tiered skip list and found that given the expected structure of this skip-list, the worst case number of nodes visited for a search operation is $\frac{n}{2} + 2$, where $n$ is the number of nodes in the original linked list. Suppose that while constructing the second tier, the probability that any node will be duplicated in the second tier from the tier below is $\frac{1}{3}$ (i.e. For any key in tier 1, the key will appear in tier 2 with probability $\frac{1}{3}$). Give the worst case number of nodes visited for a search operation on the expected structure of such a list.

- $\frac{n}{3} + 3$

- $\frac{n}{3} + 2$
- $\frac{2n}{3} + 2$
- $\frac{2n}{3} + 3$

### Rationale

First let's consider the expected structure we create. The bottom list has, of course, $n$ nodes in it. Since each node from the bottom list is copied into the second list with probability $\frac{1}{3}$, we calculate the expected number of nodes in the second list as follows: Define a random variable $X_i$ to be 1 if node $i$ from the bottom list is copied to the second list, and 0 otherwise. Then $Pr[X_i = 1] = \frac{1}{3}$ and $E[X_i] = 1 \cdot Pr[X_i = 1] + 0 \cdot Pr[X_i = 0] = \frac{1}{3}$. In the worst case, a search will visit all the nodes in the copied list, making for a total of $\frac{n}{3}$ node visits. We now calculate the expected number of nodes visited in the bottom list. The probability that any node is followed by $k$ nodes without duplicates is $\left(\frac{2}{3}\right)^k$, thus the expected number of nodes examined in the third phase is at most $1 + \sum_{k>0} \left(\frac{2}{3}\right)^k = 1 + \left(\sum_{k\geq 0} \left(\frac{2}{3}\right)^k - 1\right) = 1 + \left(\frac{1}{1-\frac{2}{3}} - 1\right) = 3$. Thus, in the worst case, we examine $\frac{n}{3} + 3$ nodes.

## Problem 9

Give the worst case running time on a skip-list with $n$ nodes and a constant number of levels.

- $O(n)$ ✓
- $O(\log n)$
- $O(n \log n)$
- $\infty$

### Rationale

You might think the worst case running time would be infinite since tthe skip-list is a random data structure. This would be true for actually building the skip-list, since it is possible to just continuously duplicate nodes forever. The search algorithm for skip-lists, unlike node insertion, is actually deterministic, that is, searching in and of itself contains no random elements. The skip-list given to you was of finite size ($n$ nodes and a constant number of levels), thus, in the worse case, there are very few short cuts and the number of node visits will be asymptotically linear on the size of the original linked list.

## Problem 10

Suppose for some randomized algorithm, we have shown that the expected running time for this algorithm is $O(n)$, where $n$ is the size of the input. We wish to show this expected running time will hold with "high probability". According to the definition given in the slides for this section, which of the following functions $f : N \to [0, 1]$ are examples of a "high probability" result? Select all that apply.

- $f(n) = 1 - \frac{1}{n}$
- $f(n) = 1 - \frac{1}{2^n}$
- $f(n) = 1 - \frac{1}{\log n}$
- $f(n) = 1 - \frac{1}{\sqrt{n}}$

### Rationale

According to the definition of high probability, we require that our function be at least $1 - \frac{1}{n^c}$ for some constant $c \geq 1$. This holds for the first highlighted answer, as $c = 1$. It it also holds for the second answer since $1 - \frac{1}{2^n}$ grows faster than $1 - \frac{1}{n^c}$. The remaining answers are incorrect since they both grow slower than $1 - \frac{1}{n^c}$ and reflect some $c < 1$.

## Problem 11

[Exploring the notion of high probability]

Inequality $1 + x \leq e^x$ holds for all real numbers. Suppose you have an $n$-sided die with distinct faces. Let $X$ be the number of times one must roll this die before we obtain a specific result $1 \leq r \leq n$. We know from the properties of geometric distributions, that, in expectation, we will have to roll this die $n$ times before we obtain $r$. Use the inequality above to determine the number $m$ of die rolls necessary such that the probability of rolling $r$ in any of the $m$ rolls is at least $1 - \frac{1}{n^c}$ for some constant $c$. **Select all that apply**.

- $m = n \log n$
- $m = n^2$
- $m = n$
- $m = \log n$

## Rationale

First we calculate the failure probability of rolling $r$ in any trial, or in other words, the probability of not rolling an $r$. This of course is given by $1 - \frac{1}{n}$. We then want to calculate the number $m$ of independent die rolls such that the probability of seeing an $r$ in $m$ die rolls is at least $1 - \frac{1}{n^c}$ for some constant $c$. This can be modelled as follows using the given inequality marked with a (*) since the die rolls are independent events.

$$\left(1 - \frac{1}{n}\right)^m \leq_{(*)} e^{-\frac{m}{n}}$$

Let $m = n \log n$, then

$$e^{-\frac{m}{n}} = e^{-\frac{n \log n}{n}} = e^{-\log n} = 1/n$$

Note that this is a bound on the probability of not rolling an $r$ for $m$ consecutive trials. Subtracting this result from 1 gives a lower bound on the probability that at least one of these trials yielded an $r$, thus the probability is at least $1 - \frac{1}{n}$ when $m = n \log n$. Note that letting $m = n^2$ yields a success probability of $1 - \frac{1}{e^n} \geq 1 - \frac{1}{n}$, so $n^2$ works as well.