CSE 421 Spring 2025: Midterm practice problems

Instructor: Chinmay Nirkhe
Date: May 5th, 2025 3:30-4:30pm

Introduction: The following are examples of problems you may encounter on the midterm. There are more problem here than you should expect on the actual exam. The problems on the practice exam, like the full exam, ask for only part of an algorithm, its runtime, or its proof of correctness. Do not write more than you need to. Our solutions to the practice problems are a good indicator of what we expect come the actual exam.

Lastly, the cover sheet of the midterm has been posted. Read it before you come to the exam.

Solutions

Problem 1. Circle True or False for each of the following questions for **[1 point per question]**.

1. The fastest known algorithm to compute the product of two n-bit integers runs in time $\Theta(n^2)$.

   True / [✓ False]

2. There is a polynomial time algorithm that for any undirected graph G with n vertices that has no odd cycles colors its vertices with at most 3 colors such that any adjacent pair of vertices have distinct colors.

   [✓ True] / False

3. $n^{2.1} = O(n^2 \log n)$.

   True / [✓ False]

4. If all edges in a graph have weight 1, then there is an $O(m+n)$ time algorithm to find the minimum spanning tree, where m is the number of edges and n is the number of vertices.

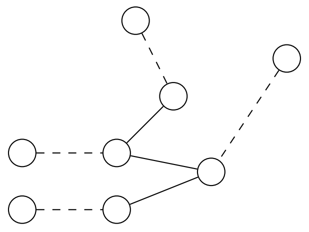
   [✓ True] / False

5. If $T(n) \leq 10T(n/3) + n^3$, $T(1) = 1$, then $T(n) = O(n^3)$.

   [✓ True] / False

Problem 2. A perfect matching of an undirected graph on $2n$ vertices is a matching of size n, namely n edges such that each vertex is part of exactly one edge. Come up with an efficient algorithm that runs in time that takes a tree on $2n$ vertices as input and finds a perfect matching in the tree, if such a matching exists.

Hint: Give a greedy algorithm that tries to match a leaf in each step. For example, in the following tree the dashed edges form a perfect matching of the given tree.



[6 points] Write down an algorithm in technical English for this problem. The algorithm must run in time poly(n). Do not prove the runtime of the algorithm.

Until there are no more edges, prune any leaf, its parent, and their neighboring edges from the tree. Record (leaf, parent).

If there are vertices remaining when all edges are removed, output that no p.m. exists.

Else, output the matching produced by the recording.

[6 points] Prove the algorithm's correctness.

Since a leaf has only 1 edge, if a p.m. exists then the edge must exist in the p.m. Furthermore, once the edge is included, neither end vertex can be included in the p.m., so the remaing p.m. exists in the pruned graph.

If a vertex has no edges, then it cannot exist in a p.m. so no p.m. exists.

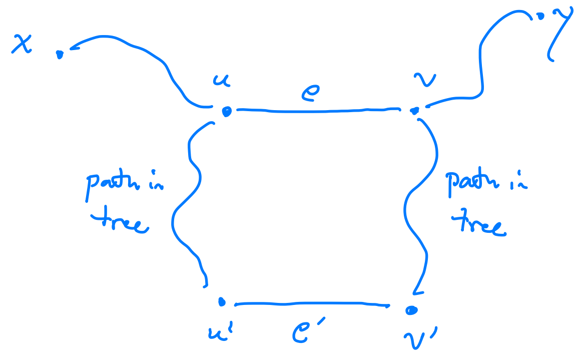
Problem 3. [10 points] Prove that in a weighted undirected graph, the heaviest edge in any cycle cannot be an edge in any minimum spanning tree.

For this problem, we say an edge e is the heaviest in a cycle if $w(e)$ is strictly greater than $w(e')$ for any other edge e' in the cycle.

Let c be a cycle and e its heaviest edge. If $e \in$ some MST then $\exists\ e' \in C \setminus \{e\}$ that is not in that MST.

Replacing e with e' will decrease the weight of the tree as $w(e) > w(e')$.

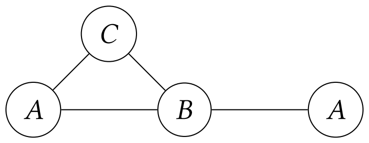
It remains to see that the replacement produces a spanning tree to complete the proof by contradiction.



Let $e = (u, v)$, $e' = (u', v')$. Then any path $x \rightsquigarrow y$ through e can be "rerouted" as $x \rightsquigarrow u \rightsquigarrow u' \longrightarrow v' \rightsquigarrow v \rightsquigarrow y$ with the paths existing from the tree.

So it is still a spanning tree after replacement.

Problem 4. Given a connected undirected graph $G = (V, E)$ with $n = |V|$ vertices and n edges. Construct an efficient algorithm to color vertices of G with 3 colors such that any two adjacent pair of vertices have distinct colors. For example, in the following graph $n = 4$ and we have colored the vertices with 3 colors.



1. **[6 points]** Let $\mathcal{A}$ be an algorithm that takes an input a connected undirected graph and outputs an edge e that belongs to a cycle (if one exists). Show how to solve the previous problem using $\mathcal{A}$ as a subroutine. Prove runtime and correctness. Your runtime statement may look something like $O(n^3) + T_{\mathcal{A}}(n/2)$ where $T_{\mathcal{A}}$ describes the runtime of the subroutine.

A n-edge connected graph is a spanning tree plus an additional edge to make one cycle. Use alg $\mathcal{A}$ to find an edge in the cycle, $e = (u, v)$.

Remove e to make a tree and color it using 2 colors since a tree is bipartite. This can be done with BFS in $O(n)$ time as $n-1$ edges.

If u, v are assigned different colors, we are done. If they are assigned the same, switch v to the 3rd color ensuring coloring of the tree and e.

Correctness was implicit in the previous statements and runtime will be

$O(n) + T_A(n-1)$.

2. **[6 points]** Give an algorithm for $\mathcal{A}$ that is as fast as possible. Do not write down the runtime or proof of correctness.

Run a variant of BFS where we output edge (u, v) if when exploring u and considering adding neighbor v, we find that v has already been visited.

Problem 5. A company buys steel rods and cuts them into smaller integer length pieces to sell. The input to the problem is a rod of steel of integer length L meters and a table of prices $p_i \in \mathbb{R}_{\geq 0}$ for $i = \{1, \ldots, n\}$ for each integer length. Construct an algorithm which calculates the optimal divisions of a rod of length L.

[10 points] Write an algorithm in technical English that computes the optimal division. Your algorithm can output a sequence of integers corresponding to the lengths of the rods. The output sequence can be in any order.

Your algorithm should be optimized for both time and space complexity. Do not give a proof of runtime or correctness. For partial credit, come up with an algorithm that outputs the value of the optimal division but not the division itself.

```
Initialize arrays $v$, next of size $L+1$ indexed from $0$ to $L$.

Set $v(0) \leftarrow 0$. next$(0) \leftarrow \perp$.

For $j$ from $1$ to $L$, calculate
    $v(j) \leftarrow \max_{i=1..n} v(j-i) + p_i$   where $v(<0) = -\infty$.
    next$(j) \leftarrow$ argmax of previous maximization.

Then, set $j = L$ and while $j - \text{next}(j) \geq 0$,
    print next$(j)$ and set $j \leftarrow j - \text{next}(j)$.
```

Problem 6. Given a *sorted* array of n *distinct* integers arranged in increasing order in $A[1, \dots, n]$, you want to find out whether there is an index i for which $A[i] = i$. Give an algorithm that runs in time $O(\log n)$ for this problem.

[2 points] Write down an algorithm for solving this problem. Do not prove runtime or correctness in this section.

```
Return $f(1, n)$ where

$f(a, b)$ for $1 \leq a \leq b \leq n$ is defined as:

    let midpt $m \leftarrow \lceil \frac{a+b}{2} \rceil$.

    If $A(m] = m$, return $m$.
    Else if $a = b$, return $\perp$.
    Else if $A(m] < m$, return $f(m+1, b)$
    Else if $A(m] > m$, return $f(a, m-1)$
```

[2 points] Prove correctness of the algorithm here.

If $A[m] = m$, then clearly a solution was found.

If $A[m] < m$, then $\forall$ $x < m$, $A[x] < x$ since A is distinct integers and $A[m] < m$. So if a sol. exists, it exists between $m+1$ and b.

Analagously, for $A[m] > m$.

Lastly if $a = b$, and $a = b = m$ is not a solution then no sol exists, by exhaustive search and prior observations.

[2 points] Prove the runtime of your algorithm here.

Notice that $\max\left\{ b-(m+1), (m-1)-a \right\} \leq \frac{b-a}{2}$

so the size halves each time.

Master theorem applied: $T(n) = T\left(\frac{n}{2}\right) + O(1)$

$\Rightarrow T(n) = O(\log n)$.