CS601 Algorithms and Complexity **2021 Jul-Nov**

Assignment 2

Total Marks: 50 *Deadline: October 24*

- Assignment needs to be done individually. If the solutions are hand-written, please make sure your writing is clear, and it is also clearly readable from the scan/image.
- Keep your answers succinct and to the point. Give clear description of your algorithms preferably in words. Use pseudocode only if really required. If you just give a pseudocode, it's highly likely that you will make mistakes. Any pseudocode needs to come with sufficient explanation, for example, what a variable is supposed to store after each iteration, how the update statement is ensuring that, what a function is supposed to compute and so on.
- You need to give arguments for correctness of your strategy/algorithm/observations/claims, whenever it is not obvious (to any fellow student).
- You are supposed to solve the assignment completely on your own, without any discussions with anyone. If you are stuck at some point, feel free to reach out to the instructor. Any immoral acts will attract severe consequences.
- You don't need to necessarily follow the given hints.

Que 1 [5+5 marks]. We are given a binary encoding scheme for an alphabet of size n. We want to design an algorithm to test whether it is uniquely decodable or not. A code is said to be **not** uniquely decodable, if there are two different strings over the alphabet whose binary encodings are same. Consider two examples below.

- $A \to 11$, $B \to 1100$, $C \to 01$, $D \to 1001$, $E \to 101$.

  This code is uniquely decodable. Different strings over $\{A, B, C, D, E\}$ are mapped to distinct binary encodings.
- $A \to 11$, $B \to 110$, $C \to 01$, $D \to 1001$, $E \to 101$.

  This is not uniquely decodable. ACD, BBC both have the same binary encoding 11011001.

Someone suggests us to build the following directed graph.

Vertices.

$V = \{w \in \{0,1\}^* : w$ is a nonempty prefix or suffix of an alphabet encoding or w is an alphabet encoding$\}$.

To elaborate, the set of vertices will be corresponding to the set of prefixes and suffixes of the alphabet encodings. An alphabet encoding of length ℓ can contribute up to $2\ell - 1$ vertices: $\ell - 1$ prefixes, $\ell - 1$ suffixes, and the encoding itself. A binary string can be a prefix/suffix for multiple alphabet encodings, but there will be only one vertex for it. For the second encoding scheme above, the vertex set will have ten vertices labeled $\{1, 11, 10, 0, 110, 01, 100, 001, 1001, 101\}$.

Edges.

$$E = \{(w_1, w_2) : w_1 w_2 \text{ is an alphabet encoding}\} \cup \{(w_1, w_2) : w_1 = \alpha w_2 \text{ for some alphabet encoding } \alpha\}.$$

To elaborate, for a vertex labeled w_1 and a vertex labeled w_2, there is a directed edge from w_1 to w_2 if and only if any of the following is true

- w_1w_2 is an alphabet encoding.
- $w_1 = \alpha w_2$ for some alphabet encoding α.

For the second encoding scheme above, for example, there will be a directed edge from 10 to 01 (because 1001 is D). There will also be a directed edge from 110 to 0 (because 11 is A). And for example, there will be no edge from 100 to 0 (because neither 1000, nor 10 is an encoding for an alphabet).

- Prove that if there is a path (of nonzero length) from a vertex labeled w_i to a vertex labeled w_j such that both w_i and w_j are alphabet encodings then the encoding scheme is not uniquely decodable.
- Prove that if the encoding scheme is not uniquely decodable then there is a path (of nonzero length) from a vertex w_i to a vertex w_j such that both w_i and w_j are alphabet encodings.

Que 2 [10 marks]. Recall the taxi scheduling problem discussed in the class. Suppose for the given set of bookings, minimum number of taxis required is k. Design an algorithm to find a 'bottleneck' of size k. That is, a set of k bookings such that no two of them can be scheduled in the same taxi. Equivalently, an independent set of size k in the given directed graph.

Hint: It is not an easy question. You might want to look at the algorithm for taxi scheduling more closely. You might want to first design an algorithm for finding a Hall's block in a bipartite graph (a subset S of left vertices with $|N(S)| = |S| - k$).

Que 3 [10 marks]. Given a set of intervals, you need to assign a color to each interval such that any two intersecting intervals should have different colors. Consider the following algorithm for this problem.

1. Initialize $c \leftarrow 1$.
2. Find a largest set of disjoint intervals (can be done via the interval scheduling algorithm).
3. Assign color c to each of these intervals and remove them.
4. If there are any intervals left then update $c \leftarrow c + 1$ and go to line 2.

Give an example where this algorithm fails to color with minimum possible number of colors. To convince the reader, please show the number of colors used by this algorithm on your example and also a better way of coloring.

Que 4 [10 marks]. There is an election with N voters and two candidates. To predict the election result, you select a sample set of k voters as follows:

```
$S \leftarrow$ set of voters
for $i = 1$ to $k$
    Choose a voter from $S$ uniformly randomly (i.e., each voter has probability $1/|S|$ of being chosen).
    Remove the chosen voter from $S$.
```

You assume that each chosen voter tells you their voting preference correctly and thus, you predict the candidate with the majority vote from the sample set as the winner.

Suppose ϵN is the winning margin for the winner candidate in the actual election. Prove that if you want your prediction to be correct with probability at least $1 - \delta$, then it suffices to take $k = O(\frac{1}{\epsilon^2}\log(1/\delta))$.

Que 5 [10 marks]. There is a processor and n jobs $\{J_1, J_2, \ldots, J_n\}$ which can be potentially scheduled on it. For $1 \leq i \leq n$, the job J_i has a processing time t_i and a deadline d_i. If you schedule the job J_i to start at time t then it will finish on time $t + t_i$. The jobs can be processed only one at a time. Your goal is to maximize the number of jobs which can be scheduled before their deadlines.

Note: Clearly, a job should either be finished before its deadline or not scheduled at all. If there is a subset of jobs which is schedulable, then their schedule can be simply in increasing order of deadlines.

Design an efficient algorithm which takes n, processing times $\{t_i\}$, deadlines $\{d_i\}$ as input and outputs the maximum number of jobs schedulable before their deadlines.

The values of processing times and deadlines are quite large, so, it is undesirable to have the algorithm running time proportional to these values. Let $T = \sum_i t_i$ and let D be the maximum deadline. Zero marks if the running time dependence is linear on T or D. For full marks, your running time should be polynomial in $(n, \log T, \log D)$. Essentially, it means that you can add/compare the processing times and deadlines. But, you cannot run a loop with D or T iterations. If you think such an algorithm is not possible, then you can write that.