CS344: Solution to Sample Final

1 Rapid Questions

1.1 Question A

False. $g(n) = \frac{2^{\log n}}{3} = \frac{n}{3} = \Theta(n) = f(n)$.

1.2 Question B

True. Given a graph with n vertices, a brute force algorithm is as follows. There are $\binom{n}{3} = O(n^3)$ subgraphs of size 3. For each subgraph, we only need to check if all 3 vertices are fully connected by 3 edges, which takes $O(1)$ time. So the algorithm takes polynomial $O(n^3)$ time.

Remark: When the clique size is $k \in \mathbb{N}^+$, the brute force algorithm is exponential in terms of n, k.

1.3 Question C

Given an undirected graph $G = (V, E)$, its complement graph is $\overline{G} = (V, \overline{E})$. Denote $VC(G, k)$ to be the problem of vertex cover of size k on G. Denote $CLIQ(G, k)$ to be the problem of finding clique of size k on G. A formal proof is to show that $VC(G, k)$ is satisfiable if and only if $CLIQ(\overline{G}, |V| - k)$ is satisfiable.

Proof. Sufficiency. If G has vertex cover $V' \subseteq V$ and $|V'| = k$, by definition we know for any edge $(u, v) \in E$, either vertex $u \in V'$ or $v \in V'$ or both. On the contrary, if there exist $u', v' \in V$ such that $u' \notin V'$ and $v' \notin V'$, then $(u', v') \notin E$ or equivalently $(u', v') \in \overline{E}$. Denote $V - V'$ to be a set of all such vertices, i.e. for any two vertices in $V - V'$, there is no edge between them in the original graph G. This implies that in the complement graph there is an edge between every pair of vertices in $V - V'$, which is a clique by definition.

Necessity. Let $V - V'$ be a clique of size $|V| - k$ in the graph $\overline{G}$. Considering an edge $(u, v) \in E$ or $(u, v) \notin \overline{E}$, we know at least one of the vertices u and v is not in the clique $V - V'$. That means at least one of u and v is in the set V'. For any such edge $(u, v) \in E$, V' must cover at least one of its vertices u or v, so V' is a vertex cover of size $|V| - (|V| - k) = k$ in G. □

1.4 Question D

True. See CLRS Corollary 22.8 (Nesting of descendants' intervals) on page 608.

2 Dynamic Programming

Let $f(i, j)$ be the max value of the optimal path from top left to cell (i, j). Since there are only 2 ways to reach cell (i, j), i.e. either from left or upper, the recurrence is straightforward:

$$f(i,j) = \begin{cases} c(i,j), & \text{if } i = 1 \text{ or } j = 1 \\ c(i,j) + \max\{f(i-1,j), f(i,j-1)\}, & \text{otherwise} \end{cases}$$

Using bottom-up DP, the running time is $O(n^2)$ and space is also $O(n^2)$.

3 Strings

1. Compute fingerprint $f(p)$ of pattern p using Rabin-Karp by setting values of all wildcards as 0. For example, given pattern "$1\phi 3$", the fingerprint is $f(p) = 1 * 10^2 + 0 * 10^1 + 3 * 10^0 \mod x$.

2. Compute fingerprints $f(t_i)$ of text $t_i = t[i \dots i + m - 1]$ using Rabin-Karp as usual.

3. For each subtext t_i, we compute some value $g(t_i)$. Suppose pattern p has q wildcards in the positions $\pi_1 < \cdots < \pi_q$ where $\pi_j \in [m]$. Then $g(t_i)$ is defined as below:

$$g(t_i) = \sum_{j=1}^{q} t[i + \pi_j - 1] * 10^{m-\pi_j} \mod x$$

4. We compute new fingerprint for each subtext as $f'(t_i) = f(t_i) - g(t_i)$ and compare it to $f(p)$.

Running time: step 1 takes $O(m)$ time; step 2 takes $O(n)$ time; step 3 takes $O(nq)$ time; step 4 takes $O(n)$ time. In total, $O(m + nq)$ time.

4 Clique

Let $f(G, k)$ be the known function to check if an undirected G has a clique of size k, and its running time is $T(G, k)$. The algorithm is described below:

```
Search-Clique(G, k)
  if not f(G,k): return none
  for v in G.V:
    G'=G\{v}
    if f(G',k): G=G'
  return G.V
```

Briefly, we can traverse each vertex in G, and see if the graph after removal of this vertex still has a clique of size k. If it is, we can safely remove this vertex since it must not belong to the final clique. Otherwise, if a vertex is in the final clique, its removal will have the remaining graph not contain the final clique.

Caveat: If there are multiple cliques in the graph, only one clique will be kept at last that is called the final clique here. That means during the iteration, one vertex to be removed may belong to some other clique, but not the final one.

Running time: the algorithm traverse each vertex in G, and at each iteration, function $f(G, k)$ is called, one vertex is removed and some edges are removed. Therefore, it costs $O(|V|T(G, k) + |E|)$ in total, which is polynomial in terms of $T(G, k), |V|, |E|$.