CS101 Algorithms and Data Structures
Fall 2022
Final Exam

Instructors: Dengji Zhao, Yuyao Zhang, Xin Liu, Hao Geng

Time: December 28th 8:00-10:00

INSTRUCTIONS

Please read and follow the following instructions:

- You have 120 minutes to answer the questions.
- You are not allowed to bring any papers, books or electronic devices including regular calculators.
- You are not allowed to discuss or share anything with others during the exam.
- You should write the answer to every problem in the dedicated box **clearly**.
- You should write **your name and your student ID** as indicated on the top of **each page** of the exam sheet.

Name	
Student ID	
Exam Classroom Number	
Seat Number	
All the work on this exam is my own. **(Please copy this and sign)**	

THIS PAGE INTENTIONALLY LEFT BLANK

1. (20 points) True or False

For each of the following statements, please judge whether it is **true(T) or false(F)**. **Write your answers in the following table**.

(a)	(b)	(c)	(d)	(e)	(f)	(g)	(h)	(i)	(j)
F	T	T	T	T	F	F	F	T	T

(a) (2') When merging two trees in the `set_union` operation of disjoint sets with union-by-height optimization, we point the root of the higher tree to the root of the shorter tree.

(a) F

Solution: We merge the shorter tree to the higher tree.

(b) (2') The adjacency matrix for a weighted undirected graph is a symmetric matrix.

(b) T

Solution: In an undirected graph, if u is connected to v, v is also connected to u. That is if u and v are connected with weight w, then $a_{uv} = a_{vu} = w$.

(c) (2') Given a connected undirected graph $G = (V, E)$ and a vertex $x \in V$. Let $E_x \subset E$ be all the edges connected to x in G. If there is an edge $e \in E_x$ whose weight is smaller than any other edge in E_x, then the minimum spanning tree of G contains e.

(c) T

Solution: Cut property.

(d) (2') Given a directed acyclic graph $G = (V, E)$ and two vertices $u,\ v \in V$. If u always appears before v in all topological sortings of G, then there exists a path from u to v in G.

(d) T

(e) (2') Dijkstra's algorithm can be viewed as a special case of A* Graph Search algorithm where the heuristic function from any vertex u to the terminal z is $h(u, z) = 0$.

(e) T

(f) (2') Floyd-Warshall's algorithm can always give the correct shortest distance between any two vertices in directed graphs with negative weights.

(f) F

Solution: Floyd-Warshall's algorithm fails when there exists negative cycle.

(g) (2') A* Graph Search algorithm returns the optimal shortest path if the heuristic function is admissible.

(g) F

Solution: The heuristic function should be consistent.

(h) (2') A Knapsack problem with $N \in \mathbb{Z}^+$ items and $W \in \mathbb{Z}^+$ capacity where the weight of each item is $w_i \in \mathbb{R}^+$ can be solved in $O(NW)$ time complexity by dynamic programming.

(h) F

Solution: Knapsack problem with $w_i \in \mathbb{R}^+$ becomes NP-Complete.

(i) (2') Any problem in P is also in NP.

(i) T

Solution: $P \subset NP$.

(j) (2') If a problem is in NP-Complete, then all the other NP-Complete problems can polynomial-time reduce to it.

(j) T

Solution: If a problem Y is NP-Complete, then $X \leq_p Y$ for any X in NP. Notice that for any problem X in NPC, X is also in NP.

2. (15 points) Single Choice

Each question has **exactly one** correct answer. **Write your answers in the following table**.

(a)	(b)	(c)	(d)	(e)
C	B	C	A	D

(a) (3') The pseudocode of `find` operation in **Disjoint Sets** is given below. Which of the following statements about the pseudocode implementation is FALSE?

Algorithm `find` Operation in Disjoint Sets

```
1: function FIND(X)
2:     if X is the root then
3:         return X
4:     else
5:         R ← FIND(Parent of X)
6:         Point X to R
7:         return R
8:     end if
9: end function
```

A. The pseudocode implementation uses path-compression optimization.

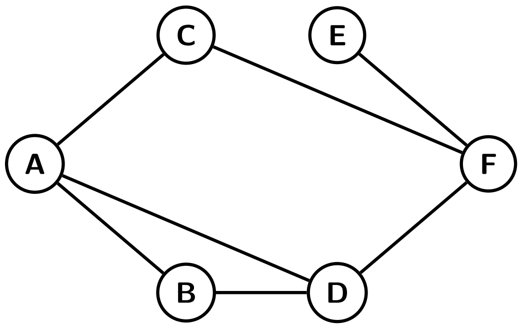
B. This function will be called at least twice in one `set_union` operation.

C. In line 3, we reach the base case where all disjoint sets are merged into one.

D. In line 6, we will let X point to the root of the set where X belongs to.

Solution:
B. `find` is called twice in a `set_union` operation. ('at least' here for recursive calls)
C. The base case is reaching the root of one tree and we cannot infer the state of other sets.

(b) (3') If we run **Depth First Traversal** on the given undirected graph, which of the following sequences is a possible order in which vertices are visited?



A. A B C D F E

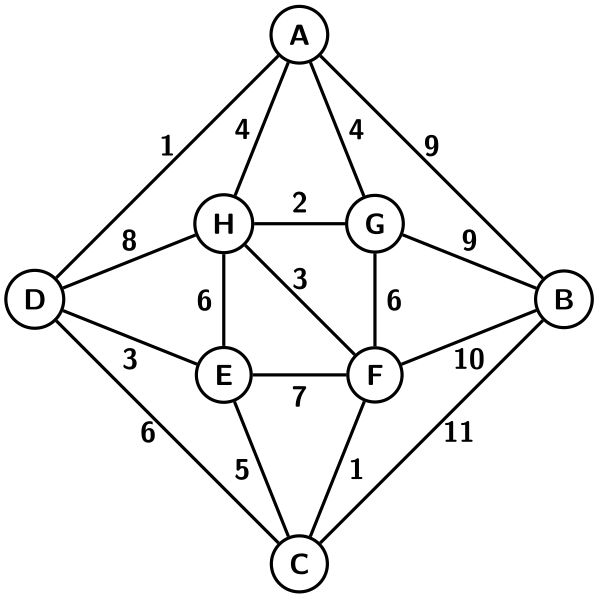
B. F C A B D E

C. B A C D F E

D. E F C D A B

Solution: A and D are BFS. C is neither.

(c) (3') For the given weighted undirected graph, which of the following statements is TRUE about its **Minimum Spanning Tree**?



A. The MST is unique, and its total weight is 23.

B. The MST is unique, and its total weight is 20.

C. The MST is not unique, and its total weight is 23.

D. The MST is not unique, and its total weight is 20.

(d) (3') Which of the following statements about **Directed Acyclic Graph** is TRUE?

A. A DAG has at least one source and at least one sink.

B. A DAG might not have a topological sorting.

C. Running topological sort in DAG takes $\Theta(|V| \log |V|)$ time.

D. A subgraph of a DAG may not be a DAG.

Solution:

A. Otherwise there is a cycle.

B. A graph is a DAG if and only if it has a topological sorting.

C. Topological sort takes $O(|V| + |E|)$.

D. The shortest path in DAG can be computed in $O(|V| + |E|)$ via topological sort. Furthermore, Dijkstra's algorithm is not applicable to DAG with negative-weighted edges.

(e) (3') Consider two problems A and B. Which of the following statements is TRUE if $A \leq_P B$?

A. If A is in NP, then so is B.

B. If A is in NP-Complete, then so is B.

C. If A can be solved in polynomial time, then so can B.

D. If A cannot be solved in polynomial time, then neither can B.

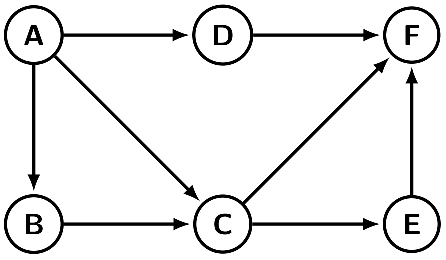
Solution:
B. B should be in NP.

3. (25 points) Multiple Choices

Each question has **one or more** correct answer(s). Select all the correct answer(s). For each question, you will get 0 points if you select one or more wrong answers, but you will get 2.5 points if you select a non-empty subset of the correct answers. **Write your answers in the following table**.

(a)	(b)	(c)	(d)	(e)
ACD	ABCD	BCD	BD	ABCD

(a) (5') Which of the following sequences is/are **Topological Sorting**(s) of the given DAG?



A. A D B C E F
B. A C E B D F
C. A B D C E F
D. A B C E D F

Solution:
All topological sortings: A D B C E F, A B D C E F, A B C D E F, A B C E D F

(b) (5') Which of the following statements about **Minimum Spanning Tree** algorithms is/are TRUE?

A. Prim's algorithm can be implemented with a binary heap data structure.
B. Kruskal's algorithm can be implemented with disjoint sets data structure.
C. When finding a minimum spanning tree, Prim's algorithm maintains a single tree structure at each iteration.
D. When finding a minimum spanning tree, Kruskal's algorithm maintains a forest structure at each iteration.

Solution: Kruskal's algorithm will maintain several disjoint trees i.e. a forest.

(c) (5') Given a directed graph $G = (V, E)$ without negative cycle where $V = \{v_1, \ldots, v_n\}$. We are interested in the shortest path from v_1 to v_n in G. Which of the following statements about **Shortest Path** algorithms is/are TRUE?

A. If the weights of some edges in E are negative, **Dijkstra's** algorithm can always find the shortest path from v_1 to v_n.
B. If the weights of all edges in E are 101, **Breadth First Traversal** can always find the shortest path from v_1 to v_n.

C. In **Bellman-Ford's** algorithm, after k out-most iterations, the shortest path from v_1 to v_n that consists of at most k edges is computed.

D. In **Floyd-Warshall's** algorithm, after k out-most iterations, the shortest path from v_1 to v_n that only allows intermediate visits to $\{v_1, v_2, \ldots, v_k\}$ is computed.

(d) (5') You are solving a **Knapsack Problem** with 4 items and capacity 5kg by bottom-up dynamic programming. The i-th item provides value v_i and weighs w_i (v_i, w_i are **positive integers** whose exact values are unknown). $\text{OPT}(i, w)$ denotes the maximum total value of a subset of items $\{1, \cdots, i\}$ with weight limit w kg and the table of computed OPT values of all subproblems is shown below. Which of the following statements is/are TRUE?

	0kg	1kg	2kg	3kg	4kg	5kg
$\varnothing$	0	0	0	0	0	0
$\{1\}$	0	0	5	5	5	5
$\{1, 2\}$	0	1	5	6	6	6
$\{1, 2, 3\}$	0	1	5	6	10	11
$\{1, 2, 3, 4\}$	0	1	5	7	10	12

A. The optimal solution achieves a total value of 12 by selecting item 1 and item 3.

B. The first item weighs $w_1 = 2$kg and the second item provides value $v_2 = 1$.

C. The possible weight of the third item w_3 is unique.

D. The possible value of the fourth item v_4 is unique.

Solution:
A. $\text{OPT}(4, 5) \neq \text{OPT}(3, 5)$, so the optimal solution must include the 4th item.
B. It is obvious that $(w_1, v_1) = (2, 5)$ and $(w_2, v_2) = (1, 1)$.
C. $(w_3, v_3) = (4, 10)$ and $(w_3, v_3) = (2, 5)$ will result in the same OPT table.
D. Determining $(w_4, v_4) = (3, 7)$ is a bit more tricky. Notice that $\text{OPT}(4, 3) \neq \text{OPT}(3, 3)$, so we can suppose $\text{OPT}(4, 3) = \text{OPT}(3, 2) + 2$, that is $(w_4, v_4) = (1, 2)$, which gives a contradiction to $\text{OPT}(4, 1) = 1 < 2$. Similarly, $(w_4, v_4) = (2, 6)$ (by guessing $\text{OPT}(4, 3) = \text{OPT}(3, 1) + 6$) is also incorrect. Therefore, $(w_4, v_4) = (3, 7)$ is the only possible option.

(e) (5') Recall that a k-COLOR problem for undirected graphs is to determine whether there exists an assignment of colors to the vertices such that no two adjacent vertices have the same color, using at most k different colors. Also recall that 3-SAT $\leq_P$ 3-COLOR and 3-SAT $\in$ NP-Complete. Which of the following statements is/are TRUE?

A. For any positive integer k, k-COLOR $\in$ NP.

B. For any problem $X \in$ NP, $X \leq_P 3$-COLOR.

C. $2\text{-COLOR} \leq_P 3\text{-COLOR} \leq_P 3\text{-SAT}$.

D. If $3\text{-COLOR} \leq_P 2\text{-COLOR}$, then $P = NP$.

Solution:

A. k-COLOR $\in$ NP since it has polynomial time certifier.

B. 3-COLOR $\in$ NP, 3-SAT $\leq_P$ 3-COLOR and 3-SAT $\in$ NP-Complete implies 3-COLOR $\in$ NP-Complete. Therefore, B is true by definition of NP-Complete.

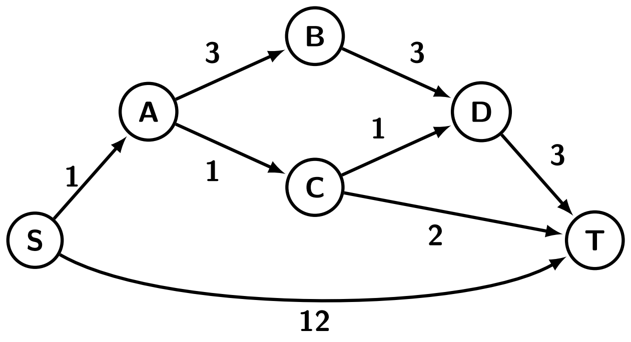
C. 2-COLOR $\in$ P, 3-COLOR $\in$ NP-Complete and 3-SAT $\in$ NP-Complete. By definition of NP-Complete, for an NP-Complete problem Y for any problem $X \in$ NP, $X \leq_P Y$. Notice that P $\subset$ NP and NP-Complete $\subset$ NP.

D. 2-COLOR $\in$ P and 3-COLOR $\leq_P$ 2-COLOR implies 3-COLOR $\in$ P. Then 3-COLOR $\in$ NP-Complete and 3-COLOR $\in$ P implies $P = NP$.

4. (7 points) Graph Algorithms Benchmark

Consider the weighted directed graph shown in the figure. For each graph algorithm listed below, write down **the order in which vertices are visited**. Assume we start from the source vertex S, and once you reach the terminal vertex T, the algorithm is ended (i.e. the order you give should be a sequence starting with S and ending with T). When choosing which vertex to visit next, always visit nodes in **alphabetical** order if there is a tie.

For Shortest Path algorithms, you should also give **the shortest path** found by the algorithm with **its length**. For A* Search algorithm, you should perform graph search and the heuristic function $h(u, T)$ is given in the table below.



u	$h(u,T)$
S	4
A	2
B	6
C	1
D	3
T	0

(a) **Graph Traversal**

i. (1') **Depth First Traversal**

Order of visited vertices: SABDT

ii. (1') **Breadth First Traversal**

Order of visited vertices: SAT

(b) **Shortest Path Benchmark**

i. **Dijkstra's Algorithm**

α) (1') Order of visited vertices: SACDBT

β) (1') Path found: SACT Path length: 4

ii. **A* Graph Search**

α) (1') Order of visited vertices: SACT

β) (1') Path found: SACT Path length: 4

γ) (1') The heuristic function $h(u, T)$ given in the table is:

- ○ consistent but not admissible.
- [✓ admissible but not consistent]
- ○ both admissible and consistent
- ○ neither admissible nor consistent

Solution: $h(u, T)$ is not consistent since $h(S, T) > w(S, A) + h(A, T)$.

5. (10 points) Cover All Points

Given a set of n points on the real axis $A = \{A_1, \ldots, A_n\}$ ($A_i \in \mathbb{R}$). Please design a greedy algorithm to find **the minimum number of unit-intervals to cover all the given points**. That is to say, each point is covered by at least one interval.

- A **unit-interval** is a closed interval $[l, r]$ where $r - l = 1$.
- A point x is **covered** by an interval $[l, r]$ if and only if $l \le x \le r$.

Example: Given $A = \{1.01, 10.1, 3.14, 2.33\}$. Then the minimum number of unit-intervals needed is 3 and $[1.0, 2.0] \cup [2.3, 3.3] \cup [9.5, 10.5]$ are 3 possible unit-intervals covering all points in A.

Note: Your solution should include the following three parts:

(a) (5') Describe your algorithm in **pseudocode** or **natural language**.

(b) (5') Proof of correctness of your algorithm using **exchange argument**.

Solution:

Algorithm Possible Greedy Algorithm

```
function FIND-UNIT-INTERVALS($A = \langle A_1, \cdots, A_n \rangle$)
    Sort $A$ in ascending order
    $I \leftarrow \varnothing$
    $R \leftarrow -\infty$
    for $i \leftarrow 1$ to $n$ do
        if $A_i > R$ then
            $I \leftarrow I \cup [A_i, A_i + 1]$
            $R \leftarrow A_i + 1$
        end if
    end for
    return $I$        ▷ The minimum number of unit-intervals needed is the length of $I$
end function
```

(1pt) Let $[l_1, r_1] \cup \ldots \cup [l_p, r_p]$ with $r_1 < \ldots < r_p$ be the greedy solution. Let $[l'_1, r'_1] \cup \ldots \cup [l'_q, r'_q]$ with $r'_1 < \ldots < r'_q$ be an optimal solution with $r_1 = r'_1, \ldots, r_k = r'_k$ for the largest possible value of k.

(1pt) Key Observation: The greedy algorithm will cover all points and place non-overlapping unit-intervals **as right as possible** i.e. $r_{k+1} \ge r'_{k+1}$. Assume $x_i \in A$ is the leftmost point on the right of r_k. Then $l_{k+1} = x_i$ and $r_{k+1} = x_i + 1$ by our greedy algorithm. Since the optimal solution should also cover x_i, $r'_{k+1} <= x_i + 1 = r_{k+1}$.

(2pt) Exchange: We can replace $[l'_{k+1}, r'_{k+1}]$ in the optimal solution with $[l_{k+1}, r_{k+1}]$. And we can exchange repeatedly for $k+1, k+2, \ldots, q-1$.

(1pt) Modification does not worsen the solution: Suppose $p > q$. There exists some point $x_j \in A$ such that $x_j = l_p$ by our greedy algorithm. But $r'_q \le r_q < l_p = x_j$, which contradicts the fact that the optimal solution should cover all points in A. Hence, $p \le q$ by contradiction. By difinition of optimal solution we have $q \le p$, and therefore $p = q$ i.e. the greedy solution is optimal.

6. (10 points) k-COLOR problem

Given an undirected graph $G = (V, E)$ and k different colors ($k \geq 4$), k-COLOR problem is to ask whether we can color the vertices so that no adjacent vertices have the same color. Recall that we have proved that 3-COLOR problem is in NP-Complete in our lecture. Now please show that k-COLOR is also in NP-Complete.

(Hint: You can recall how to prove 4-COLOR problem is in NP-Complete firstly.)

Solution:

1. First, we prove that k-COLOR $\in$ NP for each integer $k \geq 4$:

   (1pt) For any graph $G = (V, E)$ and color assignment of all vertices in V as a certificate, **there exists a poly-time certifier** which

   - enumerates all vertices in V and checks whether $\leq k$ different colors are used;
   - enumerates all edges in E and checks whether each pair of vertices have different colors.

2. Next, we prove by induction. Assume now k-COLOR $\in$ NP-Complete has been already proved, and we will show $(k+1)$-COLOR $\in$ NP-Complete still holds.

   (a) **(1pt)** From 1 we know $(k+1)$-COLOR $\in$ NP.

   (b) **(3pts)** Construction: For any k-COLOR instance with $G = (V, E)$, we add one new vertex x and construct a $(k+1)$-COLOR instance with $G' = (V', E')$ where $V' = V \cup \{x\}$ and $E' = E \cup \{\{x, v\} | v \in V\}$, which can be done in polynomial time.

   (c) Then, we show that G is k-colorable if and only if G' is $(k+1)$-colorable.

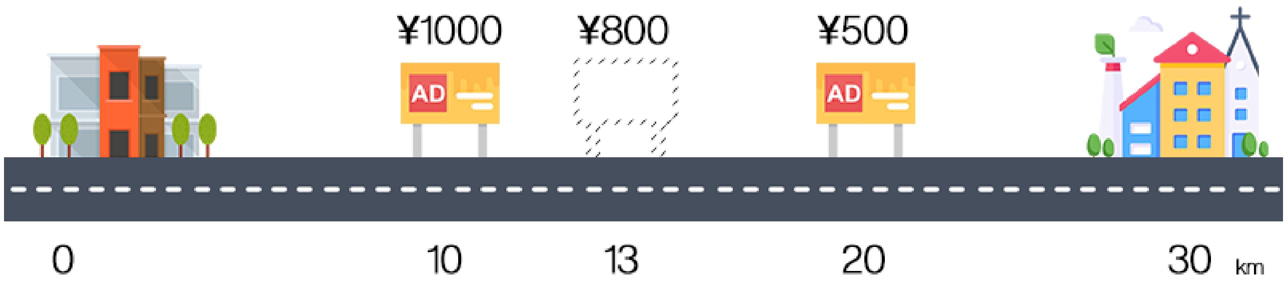
   - **(2pts)** "$\Rightarrow$": If G is k-colorable, then there exists a color assignment $c(v)$ for each $v \in V$ using colors $1, \ldots, k$. For each vertex $v \in V'$, we color $v \in V$ with $c'(v) = c(v)$ and color x with a new color $c'(x) = k + 1$. For each edge $e \in E$, if $e = \{u, v\} \in E$, $c'(u) = c(u) \neq c(v) = c'(v)$; if $e = \{x, v\} \notin E$, $c'(x) = k + 1 \neq c'(v) = c(v)$ since $c(v) \leq k$ for $v \in V$. Therefore, G' is $(k+1)$-colorable.
   - **(2pts)** "$\Leftarrow$": If G' is $(k+1)$-colorable, then x must be the only vertex in G' that has a unique color since it is adjacent to all vertices in V. W.L.O.G let $c'(x) = k+1$, and then $c'(v) \leq k$ for $v \in V$. For each vertex $v \in V$, let $c(v) = c'(v) \leq k$. Then for each edge $e = \{u, v\} \in E$, we will have $c(u) = c'(u) \neq c'(v) = c(v)$ by the fact that G' is $(k+1)$-colorable. Therefore, G is k-colorable.

3. **(1pt)** Therefore, starting with the base case that 3-COLOR problem is in NP-Complete, we can prove k-COLOR $\in$ NP-Complete by induction for each integer $k \geq 4$.

7. (13 points) Highway Billboard Schedule

There is a highway of length L kilometers connecting Alpha Town and Beta City and there are n available billboard slots along the highway. Each slot i is at $x_i \in \mathbb{R}^+$ ($0 < x_1 < x_2 < \ldots < x_n < L$) kilometers from the origin, and you will receive revenue $r_i \in \mathbb{R}^+$ if you place a billboard at slot i.

However, the highway administration department requires that the distance between two adjacent billboards should be no less than 5 kilometers. Please come up with a dynamic programming algorithm to **maximize the total revenue of placing billboards**, subject to the 5km restriction.



Example: Given $n = 3$ slots whose x_i and r_i are shown in the figure above. Then we can choose slot 1 and slot 3 to place billboards to maximize the total revenue ($1000 + 500 = 1500$). Notice that you cannot place two billboards at slot 1 and slot 2 at the same time due to the 5km restriction.

(a) (4') Define $p(i)$ as the largest index $j < i$ such that the slot j is compatible with slot i (i.e. not violating the 5km restriction). If slots $j = 1, \ldots, i-1$ are all incompatible with slot i, then $p(i) = 0$. Please design an efficient algorithm to compute all $p(i)$ for slots $i = 1, \ldots, n$ in $\Theta(n)$ runtime complexity. Describe your algorithm in **pseudocode** or **natural language**.
Hint: You could come up with a $\Theta(n^2)$ algorithm first, and then optimize the inner loop.

Solution:

Algorithm Possible Algorithm to Compute $p(i)$

```
function COMPUTE-COMPATIBLE-INDEX($X = \langle x_1, \cdots, x_n \rangle$)
    $j \leftarrow 0$
    for $i \in \{1, \cdots, n\}$ do
        while $x_i - x_{j+1} \geq 5$ do        ▷ or ($j+1 \leq n$ and $\cdots$) or ($j+1 \leq i$ and $\cdots$)
            $j \leftarrow j + 1$
        end while
        $p(i) \leftarrow j$                    ▷ or compare $x_j$ and record $j - 1$
    end for
    return $\langle p(1), \cdots, p(n) \rangle$
end function
```

Observed that $p(i)$ in non-decreasing, we can optimize the inner loop of a naive double-nested-loop structure (j is not initialized at iteration i and is kept from iteration $i-1$).

(b) (2') Define your subproblem for highway billboard schedule problem.

Solution: $\mathsf{OPT}(i)$ = maximum revenue of placing billboards at a subset of slots $\{1, 2, \cdots, i\}$

(c) (5') Give your Bellman equation to solve the subproblems.
Hint: You can directly use $p(i)$ defined in (a) and get full credits of this sub-question without answering (a) correctly. Your base case should be well-defined.

Solution:

$$\mathsf{OPT}(i) = \begin{cases} 0 & \text{if } i = 0 \\ \max\{\mathsf{OPT}(i-1),\ r_i + \mathsf{OPT}(p(i))\} & \text{if } i > 0 \end{cases}$$

Explanation: (NOT Required)

- The 1st term in max: does not place billboard at slot i
- The 2nd term in max: place billboard at slot i and get revenue r_i, with no access to incompatible slots $\{p(i)+1, \cdots, i-1\}$

(d) (2') What is the total runtime complexity of your algorithm?

Solution: $\Theta(n)$