ShanghaiTech University

CS101 Algorithms and Data Structures
Fall 2024

Homework 9

Due date: December 4, 2024, at 23:59

1. Please write your solutions in English.
2. Submit your solutions to Gradescope.
3. Set your FULL name to your Chinese name and your STUDENT ID correctly in Gradescope account settings.
4. If you want to submit a handwritten version, scan it clearly. `CamScanner` is recommended.
5. We recommend you to write in LaTeX.
6. When submitting, match your solutions to the problems correctly.
7. No late submission will be accepted.
8. Violations to any of the above may result in zero points.

1. (8 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 1 point if you select a non-empty subset of the correct answers.

Write your answers in the following table.

(a)	(b)	(c)	(d)
BC	ABD	AB	BD

(a) (2') Which of the following statements about topological sort is/are true?

A. A sub-graph of a DAG may not have a topological sorting.

B. Any directed tree has a topological sorting.

C. It's possible for a DAG to have multiple topological sortings.

D. Since we have to scan all vertices to find those with zero in-degree in each iteration, the run time of topological sort is $\Omega(|V|^2)$.

(b) (2') Which of the following statements about topological sort is/are true?

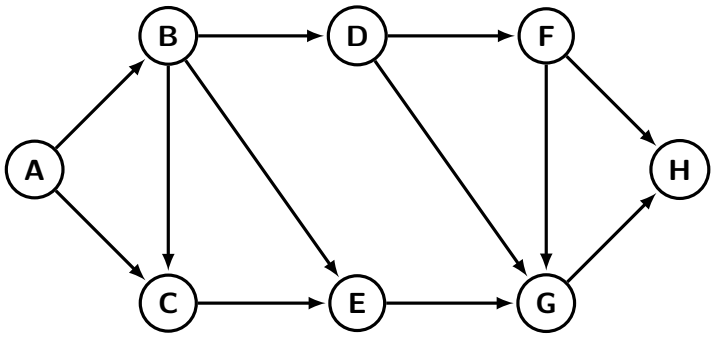
A. For a connected graph, we can determine if it has a cycle in $\Theta(|E|)$ time.

B. A critical path in a DAG is a path from the source to the sink with the maximum total weights.

C. A DAG with all different weighted edges has one unique critical path.

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

(c) (2') Which of the following sequences is/are **Topological Sorting**(s) of the given DAG?



A. A B C E G D F H

B. A B C D E F G H

C. A B D C F E G H

D. A B D E C F G H

(d) (2') For the coin-changing problem, denote

- $C = (c_1, c_2, \cdots, c_k)$: the denominations of coins, where $1 = c_1 < c_2 < \cdots < c_k$;

- X_n^*: an optimal solution, i.e., a multi-set of coins which has the minimum number of coins to make change for n;
- X_n: the greedy solution, solved by

$$X_n = \begin{cases} \{\}, & n = 0 \\ \{c_t\} \cup X_{n-c_t}, & n \geq 1, \text{where } c_t = \max_{c_i \leq n} c_i \text{ is the largest coin that can be used} \end{cases}$$

For example, if $C = (1, 2, 3)$, then X_4^* can be either $\{2, 2\}$ or $\{3, 1\}$, and $X_4 = \{3, 1\}$.

Which of the following statements is/are true?

A. If $\forall i \in [3, n], c_i = c_{i-1} + c_{i-2}$, then $\forall n, |X_n^*| = |X_n|$.

B. $\exists C, \exists i$ with $2c_i > c_{i+1}$, such that $\forall n, |X_n^*| = |X_n|$.

C. If $\forall i, 2c_i \leq c_{i+1}$, then $\forall n, |X_n^*| = |X_n|$.

D. If $\forall i \in [2, n], \frac{c_i}{c_{i-1}}$ is an integer, then $\forall n, |X_n^*| = |X_n|$.

Solution:

A. Counterexample: $C = (1, 3, 4), n = 6$, then $X_n^* = \{3, 3\}$ but $X_n = \{4, 1, 1\}$.

B. Example: $C = (1, 2, 3)$, then we can prove that $\forall n, |X_n^*| = |X_n| = \lceil \frac{n}{3} \rceil$.

C. Counterexample: $C = (1, 4, 9), n = 12$, then $X_n^* = \{4, 4, 4\}$ but $X_n = \{9, 1, 1, 1\}$.

D. Let $c_t = \max_{c_i \leq n} c_i$ be the largest denomination that can be used, and we will prove at least one c_t coin is used in the optimal solution to make change for n.

Denote x_i as the number of coins of c_i used in the optimal solution, then $n = \sum_{i=1}^{t} x_i c_i$, and we will prove that $x_t > 0$.

Note that $\forall i \in [1, t-1], x_i \leq \frac{c_{i+1}}{c_i} - 1$ in the optimal solution, because if at least $\frac{c_{i+1}}{c_i}$ coins of c_i are used, then we can replace them with one c_{i+1} coin to get a

better solution.

$$
\begin{aligned}
x_i \leq & \frac{c_{i+1}}{c_i} - 1 \\
x_i c_i \leq & c_{i+1} - c_i \\
x_t c_t = & n - \sum_{i=1}^{t-1} x_i c_i \\
\geq & n - \sum_{i=1}^{t-1} (c_{i+1} - c_i) \\
= & n - c_t + c_1 \\
= & n - c_t + 1 \\
x_t c_t + x_t \geq & n + 1 \\
x_t + 1 \geq & \frac{n+1}{c_t} > \frac{n}{c_t} \geq 1 \\
x_t > & 0
\end{aligned}
$$

2. (10 points) Test Paper Arrangement Task

Midterm week is finally over! Now the CS101 course staff is filing the test papers. Suppose there are t test papers waited to be filed, and the course staff was assigned n file cabinets with various capacities. Mark the capacity of the file cabinet as c_i $(i = 1, 2, \cdots, n)$. It's guaranteed that $\sum_{i=1}^{n} c_i \geq t$.

(a) (5') Given the number of test papers t and the capacities of each of the n cabinets $c_1, c_2, \cdots, c_n$, come up with a greedy algorithm that can file the papers in cabinets such that the least number of cabinets is used. You should give out the steps of your algorithm **(as efficient as possible)** and the time complexity (tight).

Solution:

Description:

1. Sort the cabinets in descending order of capacity.
2. While there are still papers to assign, assign the next largest cabinet to full capacity or the number of papers left (the minimum of two).

Time complexity:

Sorting the cabinets take $\Theta(n \log(n))$ time. The second step requires at most n times of checking, therefore the time complexity in all is $\Theta(n \log(n))$.

(b) (5') Prove the optimality of your algorithm using **Exchange Arguments** (any optimal solution can be iteratively exchanged to your greedy solution without making it worse). Answer the questions in step 4.5. to give your sufficient proof.

Proof:

1. Assume $c_1 > c_2 > \cdots > c_n$ after sorted. Denote a solution $X = (x_1, x_2, \cdots, x_n)$ where we put x_i test papers in the cabinet of capacity c_i. And the measurement $m(X)$ is counting how many cabinets have at least one test paper.
2. Assume the optimal solution is X^*, and the greedy solution is X.
3. Consider the first point of discrepancy between X^* and X at index i (i.e. $x_i^* \neq x_i$).
4. How to perform changes to make $x_i^* = x_i$? You should ensure that such changes is always valid in all possible scenarios of X^*.

   Solution:

5. Why such changes doesn't make X^* worse? You should consider all possible scenarios of X^*.

   Solution:

6. Continue Applying steps 3.4.5. until $X^* = X$. Finally we prove that X is an equally optimal solution.

Solution:

4. If $x_i^* = 0$, we should move all test papers from one of the non-empty cabinets among $c_{i+1}, \cdots, c_n$ to c_i. (1pt)

Then if $x_i^* \leq x_i$, we can arbitrarily move some test papers from $c_{i+1}, \cdots, c_n$ to c_i to make $x_i^* = x_i$. (1pt)

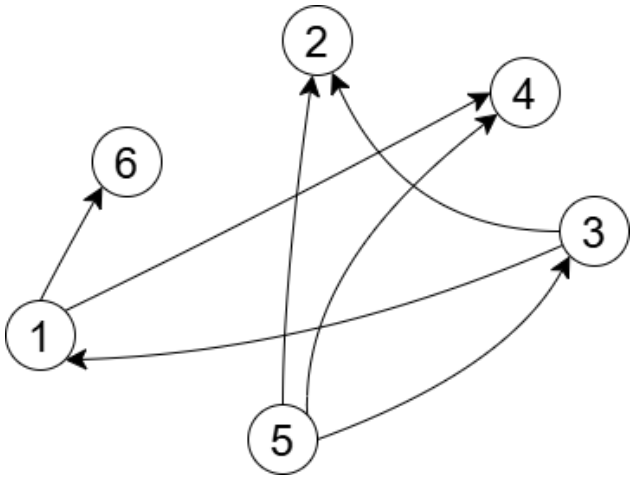
5. $x_i^* < x_i$ before we perform changes by the construction of our algorithm. (1pt)

If $x_i^* = 0$, moving all test papers from one of the non-empty cabinets among $c_{i+1}, \cdots, c_n$ to c_i doesn't change the measurement. (1pt)

Then if $x_i^* \leq x_i$, taking papers away from $c_{i+1}, \cdots, c_n$ doesn't make the measurement larger. (1pt)

3. (10 points) Topological Sorting

Consider the unweighted directed graph G below.



(a) (5') How many topological sort results are there for the graph? State all possible results and show your calculated process.

Solution: There are 8 possible topological sorts,

$$(5,3,1,6,2,4),(5,3,1,6,4,2),$$
$$(5,3,2,1,6,4),(5,3,2,1,4,6),$$
$$(5,3,1,2,6,4),(5,3,1,2,4,6),$$
$$(5,3,1,4,6,2),(5,3,1,4,2,6)$$

Vertices 5 and 3 must be the first and second vertex in any topological order since there is a directed path from 5 through 3 to every other vertex in the graph.
For the remaining four vertices, the ordering of 1, 4, and 6 is independent from 2. Vertices 1, 4, and 6 have two possible orderings: (1, 4, 6) and (1, 6, 4), while 2 can be placed in one of four positions relative to either order.

(b) (5') Construct an example of a single directed edge that, when added to G, would make it impossible to create a valid topological ordering of the graph. Then, determine the total number of such edges that could be added to G and explain why each of these edges prevents a topological ordering.
(Ensure that this edge is not a self-loop and does not duplicate an existing edge in G.)

Solution: All vertices are reachable from 5, so we can add any edge from vertices 1, 2, 3, 4, or 6 to 5 to create a cycle. All vertices except 5 are reachable from 3, so we can add any edge from vertices 1, 2, 4, or 6 to create a cycle. Vertices 4 and 6 are

reachable from 1, so we can add either edge to 1.

$$(1,5),(2,5),(3,5),(4,5),(6,4),(1,3),(2,3),(4,3),(6,3),(4,1),(6,1)$$

4. (8 points) Greedy algorithm

You are given a set of n cakes. Each cake i has:

- A **satisfaction value** $s_i \in \mathbb{N}^+$, representing the enjoyment you get from eating the cake.
- An **expiration date** $e_i \in \mathbb{N}^+$, specifying the latest day (starting from day 1) the cake can be eaten. If not eaten by day e_i, the cake is no longer valid.

You can eat at most **one cake per day**, starting from day 1, up to day $E = \max_{i \in [1,n]} \{e_i\}$ i.e. the maximum expiration date of any cake. Your goal is to determine the best strategy to eat the cakes to **maximize the total satisfaction**, defined as the sum of the satisfaction values of the cakes you consume.

To illustrate, consider the following example of $n = 9$ cakes:

i	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
s_i	10	20	10	15	14	40	18	5	20
e_i	5	3	3	3	5	4	5	7	5

The maximum expiration date is $m = 7$, so you need to decide which cake to eat on each day from day 1 to day 7. Note that:

- You can only eat cake i on day j if $j \leq e_i$, due to the expiration date constraint.
- You can leave some days without eating any cake if that results in a higher total satisfaction.

We represent a possible allocation as $C = \{c_1, c_2, \ldots, c_E\}$, where c_j is the index of the cake eaten on day j, and "-" indicates no cake is eaten on that day.

Here are three valid allocations for this example:

- $\{1, 2, 3, 5, 7, 8, -\}$, with a total satisfaction of $77 = 10 + 20 + 10 + 14 + 18 + 5$.
- $\{1, 2, 3, 6, 7, -, 8\}$, with a total satisfaction of $103 = 10 + 20 + 10 + 40 + 18 + 5$.
- $\{2, 4, 6, 7, 9, -, 8\}$, with a total satisfaction of $118 = 20 + 15 + 40 + 18 + 20 + 5$.

For this instance, one of the optimal allocations is $\{2, 4, 6, 7, 9, -, 8\}$, resulting in the maximum total satisfaction of 118.

Notes for how to answer:

For the following two greedy algorithms, you need to judge whether it always find an optimal solution.

If not always, please give a counterexample, including the input, the greedy solution and an optimal solution.

If always, please give a strict proof by **"Greedy Stays Ahead" Arguments** including:

- (1pt) **A proper definition of a series of possible sub-problem solutions** $X_1, X_2, X_3, \cdots$, based on what this algorithm produces after each iteration.

  For example, in coin changing problem, sub-problem solution X_i is a multi-set of coins whose denominations add up to i.

- (1pt) **The definition of optimality**. You should give a series of measurements on each sub-problem $m_1(X_1), m_2(X_2), \cdots$, and then define the optimal sub-problem solution X_i^* by:

  $$X_i^* \in \arg\max\{m_i(X_i)\} \text{ or } X_i^* \in \arg\min\{m_i(X_i)\}$$

  For example, in coin changing problem, $m_i(X_i) = |X_i|$ is the number of coins, and the optimal solution is the minimal one.

- (2pts) **The proof that your greedy algorithm always stays ahead by induction**. Denote X_i as the greedy sub-problem solution, and you can prove like:

  $$m_{i-1}(X_{i-1}) = m_{i-1}(X_{i-1}^*) \implies m_i(X_i) = m_i(X_i^*)$$

  And usually it is proved by contradiction. For example, prove

  $$m_{i-1}(X_{i-1}) \neq m_{i-1}(X_{i-1}^*) \impliedby m_i(X_i) \neq m_i(X_i^*)$$

(a) (4') Consider days in ascending order and choose the cake with maximal s_i first.

Algorithm 1 Days ascending, Cake maximal s_i

Require: A set of cakes, where each cake i has satisfaction s_i and expiration e_i.
Ensure: An allocation with the maximum total satisfaction.

```
C ← {−, −, ··· , −}                          ▷ Initial arrangement is empty
Sort the cakes by s_i in descending order
i ← 1                                        ▷ Consider cakes by s_i in descending order
for j = 1 to E do                            ▷ Consider days in ascending order
    while i ≤ n and e_i < j do
        i ← i + 1                            ▷ Skip expired cakes
    end while
    if i ≤ n then
        C_j ← i
        i ← i + 1
    end if
end for
return C
```

Solution:
Not always.

Counterexample 1:

Input: $(s_1, e_1) = (1, 1); (s_2, e_2) = (2, 2)$.

Greedy solution: $C = \{2, -\}$ with total satisfaction 2.

Optimal solution: $C = \{1, 2\}$ with total satisfaction 3.

Counterexample 2:

Input: The example we provided in problem description.

Greedy solution: $C = \{\}$ with total satisfaction .

Optimal solution: $C = \{\}$ with total satisfaction .

(b) (4') Consider days in descending order, and choose the cake with maximal s_i first.

Algorithm 2 Days descending, Cake maximal s_i

Require: A set of cakes, where each cake i has satisfaction s_i and expiration e_i.

Ensure: An allocation with the maximum total satisfaction.

```
C ← {−, −, ··· , −}                          ▷ Initial arrangement is empty
H ← an empty heap, where H.top() returns the index i of the cake with maximal s_i.
for j = E downto 1 do                        ▷ Consider days in descending order
    for every i such that e_i = j do
        H.push(i)                            ▷ Cake i can be eaten before day j
    end for
    if H.notEmpty() then
        C_j ← H.top()                        ▷ Consider cakes by s_i in ascending order
        H.pop()
    end if
end for
return C
```

Solution:

Always.

A proper definition of sub-problem solutions:

X_j means a possible allocation if we only eat cake during day $[j, E]$.

The definition of optimality:

$m(X_j)$ means the total satisfaction we eat in X_j.

And the optimal sub-problem solution is the largest one: $X_j^* \in \arg\max\{m(X_j)\}$

The proof that greedy stays ahead by induction:

The greedy sub-problem solution X_j is identical to C after each iteration of j in the Algorithm 2 pseudocode.

Suppose $m(X_{j+1}) = m(X^*_{j+1})$ i.e. greedy stays ahead from X_E to X_{j+1}.

If there exists a more optimal solution X'_j such that $m(X'_j) > m(X_j)$, we can lead to contradiction.

Let X'_{j_j} be the cake eaten in day j in solution X'_j.

If $X'_{j_j} \neq -$ and $X'_{j_j} \in X_{j+1}$, suppose $X'_{j_j} = X_{j+1_k}$, then we can change the solution X_j by swapping X'_{j_j} and X'_{j_k}. This change is valid, and $m(X'_j) > m(X_j)$ still holds after changing.

Continue changing until $X'_{j_j} = -$ or $X'_{j_j} \notin X_{j+1}$. Then we have $s_{X'_{j_j}} > s_{X_{j_j}}$, because $m(X'_j) > m(X_j)$ and $m(X'_{j+1}) \leq m(X_{j+1})$ because X_{j+1} stays ahead.

However, it contradicts with the fact that $s_{X'_{j_j}} \leq s_{X_{j_j}}$, because we select the cake from heap with maximal s_i in pseudocode line 8: $C_j \leftarrow H.top()$.