CS 421 Algorithms (Summer 2021)

Homework #2 (80 points)

Due Date: at noon on 7/27/2021 (Tuesday).

This homework will be discussed during the class on 7/27/2021.

This homework is a preparation for 2nd mid-term exam.

2nd mid-term exam will be taken place on 7/28 (Wednesday).

Submission Instruction:

- Convert your homework 2 to a single PDF file and the file name should be in a format using your name. For example, **JoeSmith421H2.pdf**
- Log into onyx and upload your homework 2 to an empty directory (i.e., the directory will contain only your homework 1 file).
- Within the directory, issue the following command
`submit jhyeh cs421 h2`

- **Q1(14 points): 0-1 Knapsack Problem:**

  There are 5 items and a knapsack. The knapsack can carry at most 6 pounds. The following table gives the values and weights of all items. If we would like to select items and put them

items	I_1	I_2	I_3	I_4	I_5
worth	11	8	2	5	14
weight	4	3	1	2	5

  into the knapsack so that the value of the load is maximized.

  (a)(10 points) Please construct and draw the necessary tables.

  (b)(4 points) Based on the tables you construct, what is the optimal solution (the items you picked).

- **Q2(31 points): Dynamic programming VS. Greedy Algorithm**

  A variant of the 0-1 knapsack problem is described as follows.
  Input: There are n items $\{1, 2, \ldots, n\}$. The i-th item weights w_i pounds, $1 \leq i \leq n$. The knapsack can carry at most W pounds.
  Output: We would like to choose items and put them into the knapsack so that the weight of the load is maximized.

  (a)(4 points) Does this variant of 0-1 knapsack problem have the optimal substructure property?

  (b)(10 points) If your answer to part(a) is "No", please disprove (i.e., give a counter-example) the optimal substructure property of the problem. Otherwies, please prove the following optimal substructure statement.

  `Statement:` `Let` $L = \{\ldots, j, \ldots\}$ `be an optimal load (i.e.,` $w(L)$ `is maximized) for the problem choosing from` n `items` $S = \{1, 2, \ldots, n\}$ `and with knapsack carrying at most` W `pounds. Then a load` $L' = L - \{j\}$ `must be an optimal load (i.e.,` $w(L')$ `is maximized) for the subproblem choosing from` $n - 1$ `items` $S' = S - \{j\}$ `and with knapsack carrying at most` $W - w_j$ `pounds.`

(c)(10 points) Let $m(i, j)$ denote the maximal weight of a load for a subproblem – maximizing the weight of a load by choosing items from $\{1, 2, \ldots, i\}$ and put them into a knapsack which can carry at most j pounds. Please recursively define $m(i, j)$.

$$m(i,j) = \begin{cases} \underline{\hspace{8cm}} & \text{if } i = 0 \text{ or } j = 0 \\ \underline{\hspace{8cm}} & \text{if } i, j > 0 \text{ and } w_i > j \\ \underline{\hspace{8cm}} & \text{if } i, j > 0 \text{ and } w_i \leq j \end{cases}$$

(d)(7 points) A greedy algorithm is described as follows. We first sort the items in an order of non-decreasing weights. Then, we just simply put the items, in the sorted order, to the knapsack until the knapsack cannot carry anymore items. This algorithm is incorrect. Please give a simple counter-example to disprove the algorithm.

- **Q3(20 points): Amortized cost analysis:**

  If we use two queues "data queue" and "working queue" to implement a stack s by the following way. Suppose that both queues have no size limit.

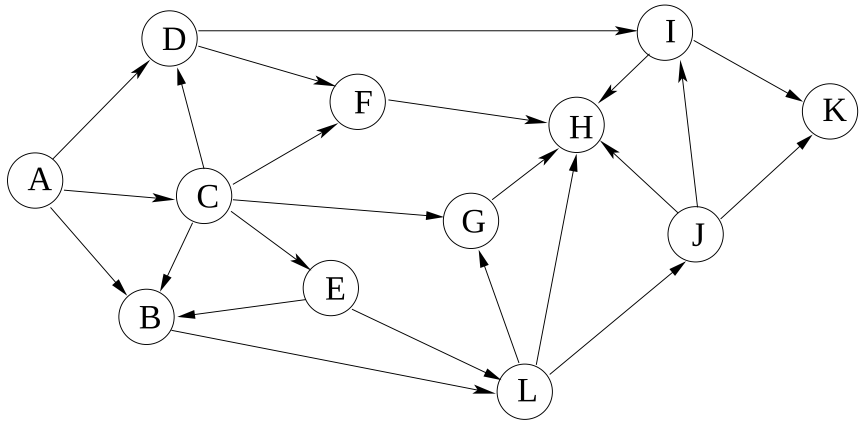
```
s.push(item)
1. enQueue the item into the data queue.

s.pop()
1. move all the items except the last one from the data queue to the working queue
2. deQueue from the data queue and return
3. now the names of the data queue and the working queue are swapped.
   That is, working queue -> data queue and data queue -> working queue
```

  (a)(10 points) Suppose that we would like to analyze the amortized costs using the accounting method. Assume there are k items in the stack before the i-th operation. If the i-th operation is a `pop` operation and we assign $2(k-1)$ amortized cost to it, then how much amortized cost for a `push` operation should be assigned? Justify your answer.

(b)(10 points) If the potential method is used, please define a potential function Φ so that the amortized costs for `push` and `pop` are constant time and linear time respectively. Assume that there are k items in the stack before the i-th operation. Justify your answer.

- **Q4(15 points): Graph Search Algorithms:**

  A weighted and directed graph is given below.



  (a)(5 points) Let vertex A be the source vertex, please find a (any) discovering sequence of vertices in a BFS search.

  (b)(5 points) Please find the discovering sequence of vertices in the DFS search, assuming that during the search if there are multiple vertices can be discovered next, please discover vertices based on their alpabetical order.

  (c)(5 points) Based on the DAG above, please find the topological sequence of vertices, assuming that during the DFS search if there are multiple vertices can be discovered next, please discover vertices based on their alpabetical order.