Homework 9 Solutions

All Sections: Due Friday, May 5 by noon on NYU Classes.
No late homeworks accepted. Contact your professor for special circumstances.

Policy on collaboration on this homework: The policy for collaboration on this homework is the same as in previous homeworks. By handing in this homework, you accept that policy. Remember: A maximum of 3 people per group.

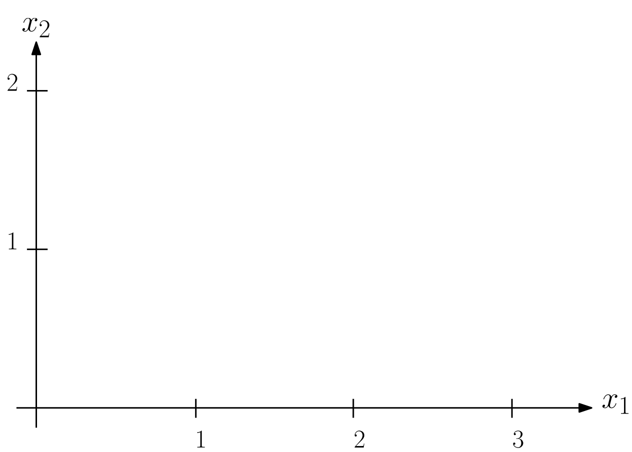
Notes: (i) Every answer has to be justified unless otherwise stated. Show your work! All performance estimates (running times, etc) should be in asymptotic notation unless otherwise noted.

(ii) You may use any theorem/property/fact proven in class or in the textbook. You do not need to re-prove any of them.

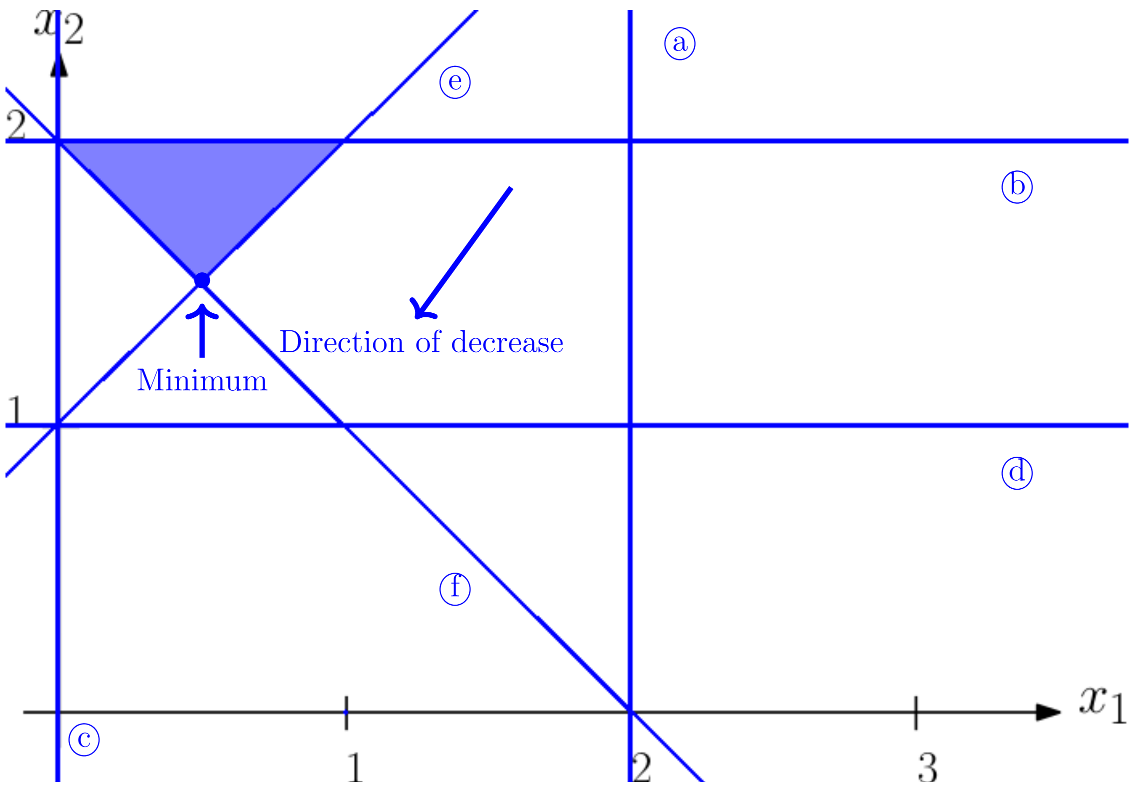
1. Consider the following linear program:

$$\begin{aligned} \text{Minimize } & 2x_2 + x_1 \\ \text{subject to } & x_1 \le 2 && \text{(a)} \\ & x_2 \le 2 && \text{(b)} \\ & x_1 \ge 0 && \text{(c)} \\ & x_2 \ge 1 && \text{(d)} \\ & x_1 - x_2 \le -1 && \text{(e)} \\ & x_1 + x_2 \ge 2 && \text{(f)} \end{aligned}$$

Draw the constraints and shade the feasible region. Label things to make it clear what is what. Mark the direction of decrease of the objective function. Mark the point realizing the minimum.



Solution:



2. Run the edit distance algorithm from section 6.3 in the book on the two strings

   `EDIT` and `DISTANCE`.

   Show the filled-out array E as in Figure 6.4(b). What is the edit distance between them?

   Solution:

		E	D	I	T
	0	1	2	3	4
D	1	1	1	2	3
I	2	2	2	1	2
S	3	3	3	2	2
T	4	4	4	3	2
A	5	5	5	4	3
N	6	6	6	5	4
C	7	7	7	6	5
E	8	7	8	7	6

The minimum edit distance between `EDIT` and `DISTANCE` is 6.

3. In the edit distance problem, we ask for the minimum number of insertions, deletions, and replacements necessary in order to turn a string $x[1 \dots m]$ into a string $y[1 \dots n]$.

   Suppose that $x[1 \dots m]$ and $y[1 \dots n]$ are both strings of decimal digits (e.g., 120421). Define a *modular increment* to be the operation of replacing a digit d by the digit $(d+1)$ mod 10.

   Define the increment-edit distance to be the minimum number of insertions, deletions, and modular increments necessary to turn $x[1 \dots m]$ into $y[1 \dots n]$. That is, it is like edit distance, except that instead of allowing arbitrary replacements, we only allow modular increments.

   Let $Iedit[i, j]$ be the minimum increment-edit distance between $x[1 \dots i]$ and $y[1 \dots j]$.

   (Note: You could in principle turn 2 into 6 by doing 4 modular increments. However, it isn't worth doing it this way, because it takes fewer steps to just delete 2 and insert 6.)

   Modify the recurrence for $E[i, j]$ to get a recurrence for $Iedit[i, j]$.

   Don't forget the base cases, $i = 0$ and $j = 0$.

   Solution: The increment-edit distance problem is virtually the same as the original edit distance problem. For the best alignment between $x[1..i]$ and $y[1..j]$ in the increment-edit distance problem, the rightmost column will be either an insertion, deletion, or a series of modular increment operations. The cost of insertion and deletion will remain the same from the original edit distance problem, but the cost of the modular increments will be different from the cost of the replacement. The cost of the modular increments from $x[i]$ to $y[j]$ can be defined as $(y[j] - x[i]) \bmod 10$, so the final recurrence including the base case is:

$$Iedit[i,j] = \begin{cases} j & \text{if } i = 0 \\ i & \text{if } j = 0 \\ min\{Iedit[i, j-1] + 1, Iedit[i-1, j] + 1, Iedit[i-1, j-1] + ((y[j] - x[i]) \bmod 10)\} & \text{otherwise} \end{cases}$$

4. You work for a television company that will be broadcasting a major sports event next year, that will last m minutes. During the broadcast, viewers will see text scrolling across the bottom of the screen advertising various products. Companies bid for the rights to place their advertisements here. A bid has the form $([a, b], x)$, where a, b, and x are real numbers, $0 \leq a < b \leq m$, and $x > 0$. It indicates that the company is offering to pay $\$x$ for the right to advertise beginning a minutes from the start of the broadcast, and ending b minutes after the start of the broadcast.

   Suppose that the television company has collected n bids,

$$([a_1, b_1], x_1), ([a_1, b_2], x_2), \dots, ([a_n, b_n], x_n)$$

   Assume that the bids are ordered in increasing order of the b_i, and for simplicity, assume that all the numbers a_i, b_i are distinct.

Your task is to design a dynamic programming algorithm that determines the maximum profit that the television station could make by accepting a subset of the bids. Since only one company can advertise at any particular time, the time periods of the accepted bids cannot overlap.

For example, if the bids are $([1, 3], 10000)$, $([2, 5.2], 50000)$, $([4, 6], 60000)$, then the maximum possible profit is \$70000, achievable by accepting the first and third bids, for ads that will run during the periods $[1, 3]$ and $[4, 6]$.

Your dynamic programming algorithm should work by filling in a table.

Hints: Think about which bids should be accepted if you are restricted to accepting bids numbered 1 through i. What should you do with the ith bid? Exploit the fact that $b_1 < b_2 < \ldots < b_i$.

To present your dynamic programming algorithm, answer the following questions:

(a) What are the dimensions of the table?

Solution: The table is a one-dimensional array of size $n + 1$, assuming we are indexing starting from 0.

(b) What is stored in table?

Solution: $T[i]$ stores the maximum profit using the first i bids

(c) What is the recurrence we will use to fill the table?

Solution:

$$T[i] = \begin{cases} 0 & \text{if } i = 0 \\ max\{T[i-1], T[k] + x_i\} & \text{otherwise} \end{cases}$$

where k is the highest index such that $b_k < a_i$

(d) What is the idea behind the recurrence? Explain in English.

Solution: When we are introduced with a new bid, there are two choices we can make for it. We either take it or we don't take it. If we do take it, then our maximum profit will be the maximum profit using the bids that don't overlap plus the profit of the new bid, this is $T[k] + x_i$ in the recurrence. If we don't take it, then our maximum profit will be the maximum profit of the previous bids that came before it, this is $T[i-1]$ in the recurrence. The bigger of these two values will provide the maximum profit using the bids up to i.

(e) What order is used to fill in the table?

Solution: From the recurrence, the indices less than i need to be filled in before $T[i]$ can be filled in, so the table should be filled in from left to right, or from 0 to n.

(f) Once the table is filled in, what value is returned as the answer to the problem?

Solution: $T[n]$ is returned. It holds the maximum profit using the first n bids, which is the entire set of bids.

The full psuedocode:

```
T[0] = 0
for i = 1 to n:
    k = i
    while k > 0 and b_k > a_i:
        k = k - 1
    T[i] = max{T[i - 1], T[k] + x_i}
return T[n]
```

5. Restate the following problem as an LP (linear programming) problem. Do not forget (a) list the variables, (b) list ALL the constraints, and (c) list the objective function and whether you are minimizing or maximizing it.

   You are planning a schedule of activities for a group of friends: Alexa, Bob, Carol, and Dudley. The possible activities are Sight-Seeing, Fishing, Running, Hacking, and Karaoke. The friends will do all activities together.

   Here we list the activities that each person likes:

   - Alexa: Fishing, Hacking, Sight-Seeing
   - Bob: Fishing, Running, Karaoke
   - Carol: Fishing, Hacking
   - Dudley: Hacking, Sight-Seeing, Karaoke

   Here is the cost of each activity, per hour:

   - Sight-Seeing: \$6
   - Fishing: \$5
   - Running: \$3
   - Hacking: \$1
   - Karaoke: \$7

   Determine how many hours the group should spend doing each activity. The number of hours of activity does not have to be an integer. For example, doing $\frac{1}{2}$ hour of fishing, $\frac{1}{4}$ hour of running, and $\frac{1}{4}$ hour of karaoke will cost $\frac{1}{2} \cdot 5 + \frac{1}{4} \cdot 3 + \frac{1}{4} \cdot 7 = 2.5 + 0.75 + 1.75 = 5$ dollars total.

   Minimize the total cost of the activities, while also making sure that each person spends at least 1 hour doing activities that he/she likes. (A person need not spend 1 hour on the same activity. For example, a person may spend a half hour on each of two different activities they enjoy.)

   You do NOT have to solve the LP. Just write it down.

Solution: We create a variable for each activity:
S = Sight-Seeing
F = Fishing
R = Running
H = Hacking
K = Karaoke

Each person introduces a constraint as we want each person spending at least 1 hour doing activities they like:

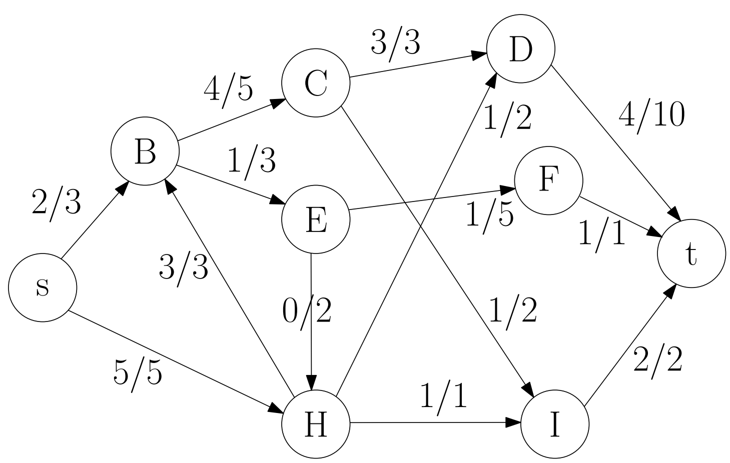
Alexa: $F + H + S \geq 1$
Bob: $F + R + K \geq 1$
Carol: $F + H \geq 1$
Dudley: $H + S + K \geq 1$

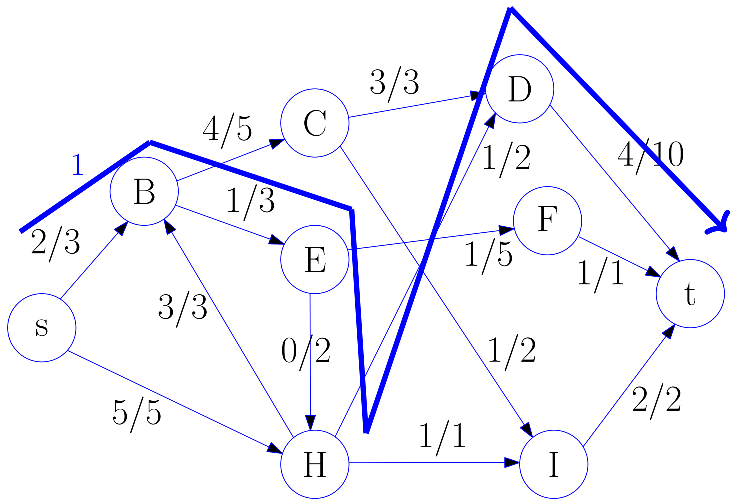
We also need to make sure the time spent doing each activity is not negative:
$S \geq 0$
$F \geq 0$
$R \geq 0$
$H \geq 0$
$K \geq 0$

The objective function to minimize is the cost of all activities:
Minimize $6S + 5F + 3R + 1H + 7K$

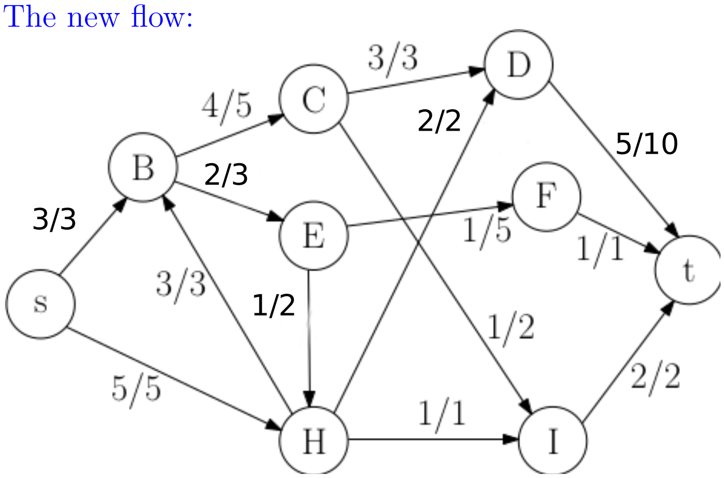
6. **Aronov's Section Only:** Consider the following flow network, with edges labeled by the usual convention: 8/10 on an edge means the capacity is 10 and the flow is 8.

   (a) Draw the residual network. As in one round of Ford-Fulkerson algorithm, *EITHER* find an augmenting path (a way to increase the flow) *OR* show a saturated cut.





Solution:
The path $s \to B \to E \to H \to D \to t$ can be augmented
The new flow:



(b) *If* you do find an augmenting path in (a), show the improved flow. Then do (a) *one more time* on the improved flow: Find an augmenting path or show the saturated cut. Stop.

Solution: Saturated cut: