CS577 Final Exam Yizhou Liu liu773@wisc.edu.

1. a) Definition: $E(i)$ represents the expected total points the player gets at level i.

Recursion Equation: $E(i) = \max_{\substack{j \\ 1 \le j \le m}} \sum_{k=i+1}^{n} [P_{ij} + E(k)] \pi_{ijk}$.

< for $k \le j \le m$, action j has a probability of π_{ijk} to bring player to level k > $(i \le k \le n)$

Base Case: $E(n) = 0$.

Final Solution: return $E(0)$.

b) Starting from the base case (ie, $E(n)=0$), for every level i, there is always an action (ie, j) that can maximize the expected total points. By induction, we can always ensure that the recursive equations will give us the optimal result.

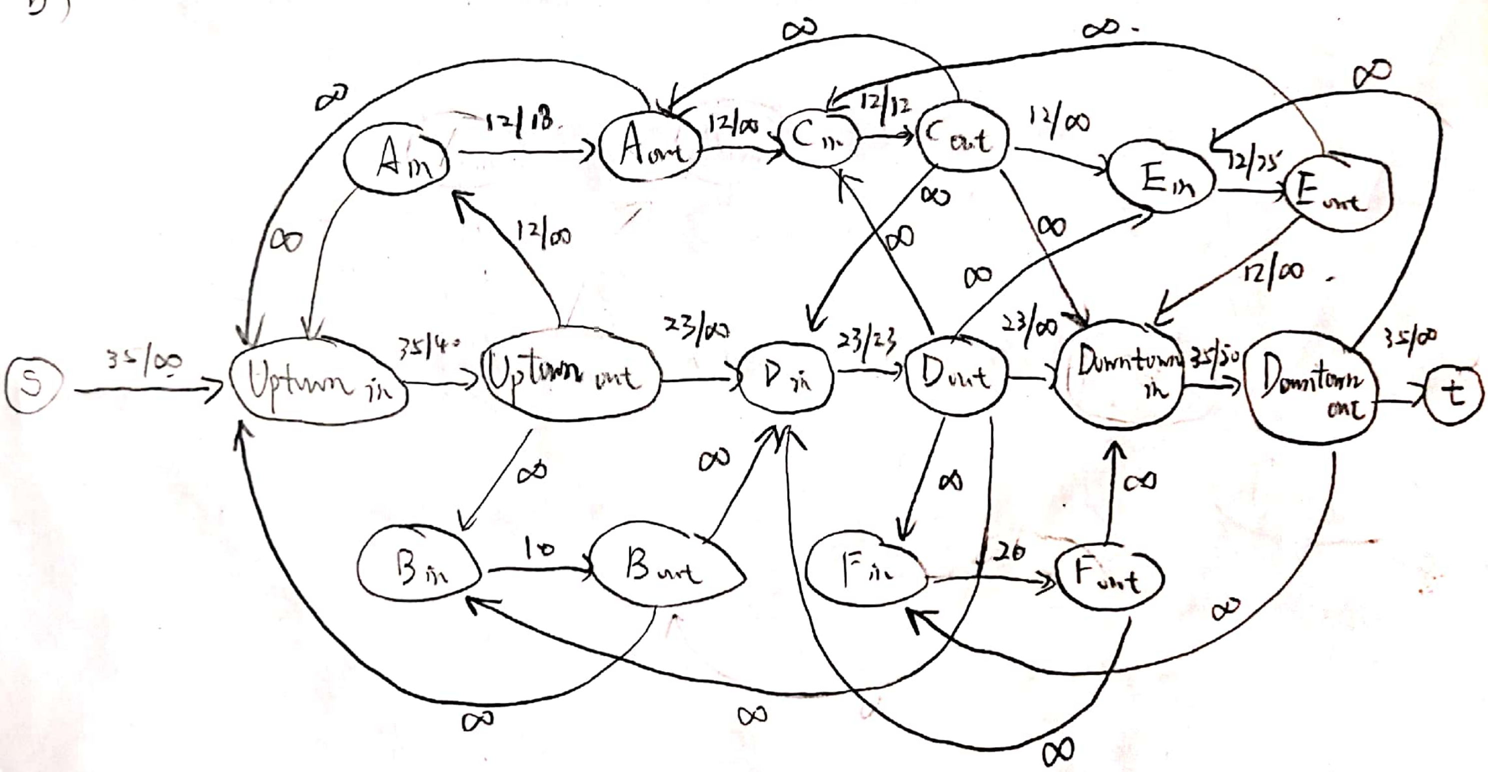
c) Algo:

- Construct array mem[n] (store optimal results)
- Construct array exp[n] (store $E(i)$).
- Set exp[n] = 0.
- for $i \leftarrow n-1$ to 1:
  - calculate $E(i) = \max \sum_{k=i+1}^{n} (P_{ij} + E(k)) \cdot \pi_{ijk}$ (k is the random level we get at level i).
  - Store the results in mem[] and exp[] arrays
- Return mem[0] (optimal expected total points)

d) $O(mn^2)$.

2. a) We reduce the problem into a max s-t flow one. First, we convert every edge on the graph (ie, $(u, v \in E)$) to directed edges with ∞ capacity. Then, we use a pair of vertices (noted as X_{in}, X_{out}, for example) to denote every vertex $v \in V$ in the given graph. We define X_{in} as the incoming vertices, incoming all the incoming edges of X go into X. And X_{out} means that all the out-going edges of X go out of (start from) X. We add edges (X_{in}, X_{out}) with a capacity of cost of X. Next, we insert source S and sink t into our graph (ie, s to uptown, t to downtown, respectively). The capacity of these edges is ∞. Finally, we perform Ford-Falkerson algorithm to get the max s-t flow f^*, which gives us the minimum cost as our final answer.

b)



Network flow instance is given as above.

And optimal cost is going to be $12 + 23 = 35$.

c). Correctness:

In order to prove the correctness of reduction, we need to show that ① if we get the max s-t flow (k) of the graph, we can get the optimal cost (k) from the graph. ② if the optimal cost (k) is given, then we can ensure that it can be derived from max s-t flow (value k). To prove case ①, given that, all the edges' capacity is either ∞ or cost of ads at that station, by the max-s-t flow, we can get the min-cut for this graph. Since the sum of capacities of edges (which intersect with min-cut) is the smallest, then we can get the optimal solution in this graph. → For ②, suppose we've already got the optimal cost k for displaying the ads, since we started with the budget of k, all capacity constraints in network are satisfied, and we can send k units of flow from source s to sink t. Then, by the construction of the flow, we can get the max s-t flow of value k. Therefore, by proving ① & ②, the reduction is correct.

3. a) Reduce the knapsack problem to problem described in the question (call it as "Utility Problem")

(ie, knapsack $\leq_M$ Utility Problem)

b) → We reduce knapsack $(\{v_i\}_{i=1}^{n}, \{w_i\}_{i=1}^{n}, T, A)$ to Utility $(\{v_i - w_i\}_{i=1}^{n}, \{\phi\}_{i=1}^{n}, T, A-T\}$.

→ Notice that in knapsack, $\{v_i\}_{i=1}^{n}$ denotes value, $\{w_i\}_{i=1}^{n}$ denotes weights, T denotes capacity constraint, A denotes target value.

→ Notice that in Utility, $\{v_i - w_i\}_{i=1}^{n}$ denotes value, $\{0\}_{i=1}^{n}$ denotes prices, T denotes discount threshold, A - T denotes utility target value

扫描全能王 创建

c) There are 2 implications we need to prove for correctness:

① ⇒. We need to prove if there is a solution subset ($S \subseteq \{1 \dots n\}$) from knapsack, then it's also a solution from Utility Problem. Given that weights $W_{i\,(i \in S)}$ is less than capacity constraint T and values $V_{i\,(i \in S)}$ greater than target A, we are asked to compute the maximum values (corresponds to utility). By calculation, the utility value is greater than its target value (ie, $V_i - W_{i\,(i \in S)} \geq A - T$). Therefore, we state that ① is correct.

② ⇐. We need to prove if there is a solution subset ($S \subseteq \{1 \dots n\}$) from utility problem, then it's also a solution from knapsack. Given that the difference between the values and prices in Utility Problem is greater than its utility target value (ie, $V_i - W_{i\,i \in S} \geq A - T$). By calculation, we get that on knapsack, value is greater than the target value ($V_{i\,(i \in S)} \geq k$), weight is less than the capacity ($W_{i\,(i \in S)} \geq T$). This satisfies the condition requirements for knapsack. Therefore, we state that ② is correct.

In conclusion, by ① & ②, we've proved that whole mapping reduction is correct.