CS 660, HOMEWORK 4 (DUE: SUNDAY NOV 3, 11:59 PM)

INSTRUCTOR: HOA VU

Each question is worth 25 points. The extra credit question is worth 20 points.

When you are asked to design an algorithm, do the following: a) describe the algorithm, b) explain (or more rigorously prove) why it is correct, and c) provide the running time.

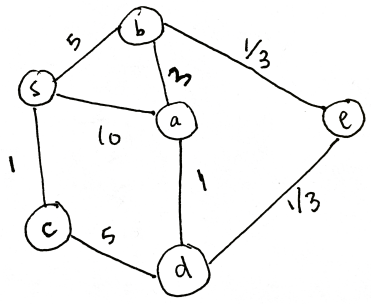
Unless specified otherwise, the problems are from the textbook by Jeff Erickson that we use.

For dynamic programming questions. Make sure a) you define the dynamic programming table, b) what is the recursive structure (how to fill an entry of the table), c) boundary cases & the filling order, d) how to get the answer from the table, and e) state the running time.

(1) **Question 1:** Recall the algorithm $Dijkstra(G, s)$:
- For all $u \in V \setminus \{s\}$, set $dist(u) \leftarrow \infty$.
- $dist(s) \leftarrow 0$.
- $R = \{\}$.
- While $R \neq V$:
  - Pick $u \notin R$ with smallest $dist(u)$, then $R \leftarrow R \cup \{u\}$.
  - For all vertices $uv \in E$, if $dist(v) > dist(u) + \ell(uv)$, then update

$$dist(v) \leftarrow dist(u) + \ell(uv).$$

Consider Dijkstra's algorithm for finding single-source shortest paths starting from vertex s to all other nodes.



Part a) What are the different (distinct) values of $dist(a)$ and $dist(d)$ during the execution of Dijkstra's algorithm (list their values over time) in the graph above.

Part b) Let us prove the Dijkstra's algorithm correctness. We want to prove that each time we set $R \leftarrow R \cup \{u\}$, then $dist(u) = \delta(u)$. We use $\delta(u)$ to denote the length of the shortest path from s to u. In particular, when u is added to R, we have $dist(u)$ correctly computed as the length of the shortest path from s to u. Use the below to write out the complete proof.

- The base case holds, i.e., the algorithm will first adds s to R (why?) and $dist(s) = 0$ which is obviously correct.
- Induction hypothesis (I.H.): Suppose it is true that for the current set R for all $v \in R$ we have $dist(v) = \delta(v)$. Now the algorithm adds u to R, i.e., $R \leftarrow R \cup \{u\}$. We want to show that $dist(u) = \delta(u)$ at the time u is added to R. Therefore, the invariant holds every time we add a new vertex to R.
- **Consider R right before we add u to R.** Let the shortest path from s to u be Q. For the sake of deriving a contradiction, suppose $dist(u) > \ell(Q)$.
- Why there must be an edge xy on the path Q where $x \in R$ and $y \notin R$?
- Let Q_x be the part of Q that is from s to u. Why must $\ell(Q_x) + \ell(xy) \leq \ell(Q)$?
- Why must $dist(x) \leq \ell(Q_x)$ (In fact, you can also claim that $dist(x) = \ell(Q_x)$. Hint: use the I.H. and the fact that $x \in R$)? Conclude that $dist(x) + \ell(xy) \leq \ell(Q)$.
- Why must $dist(y) \leq dist(x) + \ell(xy)$ (Hint: look at the behavior of the algorithm when it added x to R)?
- Why must $dist(u) \leq dist(y)$ which is a contradiction (Note that both u and y are currently not in R)?
- Derive that $dist(u) < dist(u)$. Conclude that $dist(u) = \delta(u)$.

(2) **Question 2:** Given a 2-dimensional array $A[1 \dots n][1 \dots n]$ of numbers. Design an algorithm that rearranges the entries in A to maximize the sum $\sum_{i=1}^{n} \sum_{j=1}^{n} (i + j)A[i][j]$. Make sure you prove that your algorithm maximizes the described sum.

(3) **Question 3:** Problem 1a page 268. For simplicity, assume all weights are distinct.

(4) **Question 4:** Problem 19a page 253. Your algorithm should run in $O(|V| \log |V| + |E|)$ time. You must argue the correctness and why the running time is as claimed.

(5) **Extra credits:** Problem 5a page 269. The update should run in $O(|V|)$ time. Justify your answer.