University of Iowa | Homework 6 | CS 3330
Department of CS | Fall 2024

Assigned: Friday 11:59 PM, October 18, 2024
Due: Friday 11:59 PM, October 25, 2024

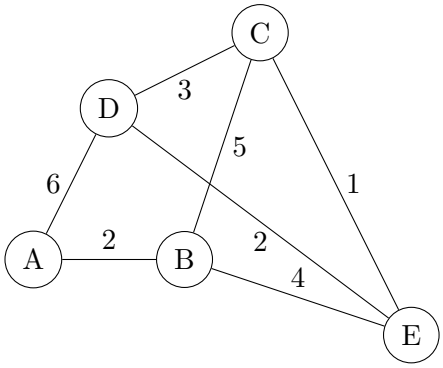
1. **[A Human Compiler]** A computer scientist must understand how algorithms work. The following algorithm is Dijkstra's algorithm, which finds the minimum distance between the source vertex s and all other vertices in the graph that are reachable from s. Execute the above algorithm for each

```
Function DIJKSTRA (starting vertex s)
    foreach v ∈ V do
        π[v] ← ∞;
        par[v] ← null;
    end
    π[s] ← 0;
    pq ← empty binary heap priority queue;
    foreach v ∈ V do
        INSERT(pq, v, π[v]);
    end
    while pq is not empty do
        u ← DEL-MIN(pq);
        foreach neighbor v of u do
            if π[v] > π[u] + cost[u → v] then
                DECREASE-KEY(pq, v, π[u] + cost[u → v]);
                π[v] ← π[u] + cost[u → v];
                par[v] ← u;
            end
        end
    end
```

of the following cases and answer the questions.

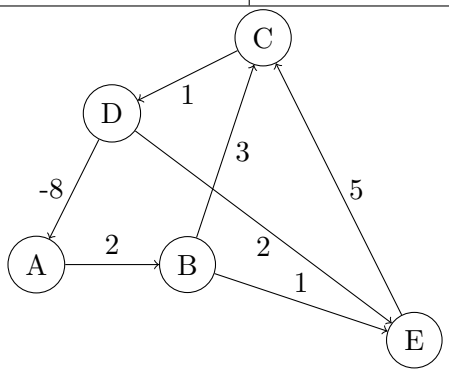
(a) **[15 points]** Display the values of the vectors π and par at the beginning of each while-loop iteration. Are the values displayed equal to the minimum distance from the source vertex? Justify your answer.

Iteration #	$\pi = (\pi[A], \pi[B], \pi[C], \pi[D], \pi[E])$	$par = (par[A], par[B], par[C], par[D], par[E])$
1	[∞,0,∞,∞,∞]	[null,null,null,null,null]



(b) **[15 points]** Display the values of the vectors π and par at the beginning of each while-loop iteration. Are the values displayed equal to the minimum distance from the source vertex? Justify your answer.

Iteration #	$\pi = (\pi[A], \pi[B], \pi[C], \pi[D], \pi[E])$	$par = (par[A], par[B], par[C], par[D], par[E])$
1	[∞,0,∞,∞,∞]	[null,null,null,null,null]



2. **[20 points] [A Safe City!]** Working in large cities at night can be daunting; you must always stay vigilant to avoid potential dangers. It's also costly, making food delivery an easy gig to earn some extra money.

   Suppose the city map is represented by a graph G, where each street is an edge (u, v) between two vertices representing locations. Panucci's Pizzeria is the starting points for all deliveries, and we need to dispatch m delivery drivers to m different destinations on the map.

   Given historical crime rate data, the probability of safely passing through a street (u, v) is given by $p(u, v)$. The safety of different streets is independent of each other, so the probability of safely passing through a sequence of streets $(u_1, u_2), (u_2, u_3), \ldots, (u_{k-1}, u_k)$ is the product of the individual probabilities:

$$p(u_1, u_2) \times p(u_2, u_3) \times \cdots \times p(u_{k-1}, u_k)$$

Design an algorithm that maximizes the probability of safe delivery for each pizza to its respective destination. The algorithm should find the safest path for each delivery person, starting from Panucci's Pizzeria, ensuring that the probability of safe passage is maximized for each route. (Hint: Consider modifying Dijkstra's algorithm.)

3. **[20 points]** **[Nana's Inheritance!]** Observing fluctuations in the stock market, one might ponder: "If I had invested my grandmother's inheritance in NVDA instead of INTC on date i and sold it on date j, I could have tripled my money!" As computer scientists, we can find an optimal algorithm to maximize this regret by revealing how many times such an opportunity occurred.

   Suppose the price of NVDA over the last n days is given by $(s_1, s_2, \cdots, s_n)$, where s_i represents the price on the $(n-i)$-th day before the present day. Develop an algorithm using the divide-and-conquer paradigm to determine the number of opportunities where one could have at least tripled their investment. Specifically, count the number of pairs (i, j) with $i < j$ where $3s_i < s_j$. What is the complexity of your algorithm?

4. **[A Human Compiler]**: A computer scientist must understand how algorithms work. Compile each of the following algorithms.

   (a) **[15 points]** Execute the "Merge-Sort algorithm" on the given array, showing the input array and the output array for each algorithm call. (The order of calls matters! See page 7 of 05DivideAndConquerI for reference.)

	Input Array				Output Array			
1st Call	-1	5	-7	0	-7	-1	0	5

   (b) **[15 points]** Execute the "Sort-And-Count algorithm" on the given array, showing the input array and the output array for each algorithm call. (The order of calls matters! See page 22 of 05DivideAndConquerI for reference.)

	Input Array				Output Array				
1st Call	5	3	-1	4	(4,	-1	3	4	5)