Topic 18 Single Source Shortest Paths

CLRS Ch 24.1-24.3

Shortest Path Property

Triangle Inequality
For any edge $(u,v) \in E$, we have $\delta(s,v) \leq \delta(s,u) + w(u,v)$.

Upper-Bound Property
We always have $v.d \geq \delta(s,v)$ for all vertices $v \in V$, and once $v.d$ achieves the value $\delta(s,v)$, it never changes.

No-Path Property
If there is no path from s to v, then we always have $v.d = \delta(s,v) = \infty$

Convergence Property
If $s \leadsto u \rightarrow v$ is a shortest path in G for some $u, v \in V$, and if $u.d = \delta(s,u)$ at any time prior to relaxing edge (u,v), then $v.d = \delta(s,v)$ at all times afterward.

Path-Relaxation Property
If $p = \langle v_0, v_1, \ldots, v_k \rangle$ is a shortest path from $s = v_0$ to v_k, and we relax the edges of p in the order $(v_0, v_1), (v_1, v_2), \ldots, (v_{k-1}, v_k)$, then $v_k.d = \delta(s,v_k)$. This property holds regardless of any other relaxation steps that occur, even if they are intermixed with relaxations of the edges of p.

Predecessor-Subgraph Property
Once $v.d = \delta(s,v)$ for all $v \in V$, the predecessor subgraph is a shortest-paths tree rooted at s

Bellman's Ford

```
BELLMAN-FORD(G, w, s)
INITIALIZE-SINGLE-SOURCE(G, s)
for i = 1 to |G.V| - 1
    for each edge (u, v) ∈ G.E
        RELAX(u, v, w)
for each edge (u, v) ∈ G.E
    if v.d > u.d + w(u, v)
        return FALSE
return TRUE
```

- allows negative weights
- brute force strategy
- computes v.d and v.π , $\forall v \in V$
- Runtime is O(VE)

Shortest Path in DAG Single Source Algorithms

```
DAG-SHORTEST-PATHS(G, w, s)
topologically sort the vertices of G
INITIALIZE-SINGLE-SOURCE(G, s)
for each vertex u, taken in topologically sorted order
    for each vertex v ∈ G.Adj[u]
        RELAX(u, v, w)
```

- vertices occurs on shortest path in order with topological sort
- run time is $\Theta(V + E)$

Dijkstra

```
DIJKSTRA(G, w, s)
INITIALIZE-SINGLE-SOURCE(G, s)
S = ∅
Q = G.V
while Q ≠ ∅
    u = EXTRACT-MIN(Q)
    S = S ∪ {u}
    for each vertex v ∈ G.Adj[u]
        RELAX(u, v, w)
```

- uses priority queue
- greedy
- no negative weights
- O((V+E)lg V)
- connected: O(E lg V)
- O(V lg V + E)

Topic 19: All Pair Shortest Path

CH. 25
Iterated Bellman Ford

- cost: $O(V^2E)$
- dense graph: $O(V^4)$
- works on graph with negative edge

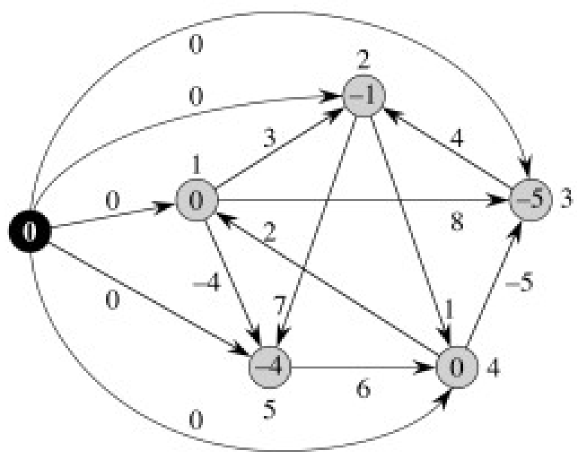
Iterated Dijkstra

- $|V|$ iteration gives $O(VElgV)$
- Dense Graph: $O(V^3lgV)$
- w Fibonnaci Heaps: $O(V^2lgV + VE)$
- does not work with negative weights

Johnson's Algorithm

```
JOHNSON(G, w)
compute G', where G'.V = G.V ∪ {s},
    G'.E = G.E ∪ {(s, v) : v ∈ G.V}, and
    w(s, v) = 0 for all v ∈ G.V
if BELLMAN-FORD(G', w, s) == FALSE
    print "the input graph contains a negative-weight cycle"
else for each vertex v ∈ G'.V
        set h(v) to the value of δ(s, v)
            computed by the Bellman-Ford algorithm
    for each edge (u, v) ∈ G'.E
        ŵ(u, v) = w(u, v) + h(u) − h(v)
    let D = (d_uv) be a new n × n matrix
    for each vertex u ∈ G.V
        run DIJKSTRA(G, ŵ, u) to compute δ̂(u, v) for all v ∈ G.V
        for each vertex v ∈ G.V
            d_uv = δ̂(u, v) + h(v) − h(u)
    return D
```

Example



- Define h(v) = $\delta(s, v), \forall v \in V$
- $\Theta(V)$ to find G'
- O(VE) to run Bellman Ford
- O(V) to find h(v)
- $\Theta(E)$ to compute w
- $\Theta(V^2)$ to initialize D
- O(V E lg V) to run Dijkstra
- Overall Runtime: O(V E lg V)
- With Fibonnaci Heaps: O(V^2 lg V + V E)
- Works with negative weights

Floyd Warshall

$$w_{ij} = \begin{cases} 0 & \text{if} i = j \\ w(i,j) & \text{if} i \neq j, (i,j) \in E \\ \infty & \text{if} i \neq j, (i,j) \notin E \end{cases}$$

FLOYD-WARSHALL′(W)

```
n = W.rows
D = W
for k = 1 to n
    for i = 1 to n
        for j = 1 to n
            d_ij = min(d_ij, d_ik + d_kj)
return D
```

Runtime: $O(V^3)$
Works with negative weight edges

$$D^{(0)} = \begin{pmatrix} 0 & 3 & 8 & \infty & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & \infty & \infty \\ 2 & \infty & -5 & 0 & \infty \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix} \quad \Pi^{(0)} = \begin{pmatrix} \text{NIL} & 1 & 1 & \text{NIL} & 1 \\ \text{NIL} & \text{NIL} & \text{NIL} & 2 & 2 \\ \text{NIL} & 3 & \text{NIL} & \text{NIL} & \text{NIL} \\ 4 & \text{NIL} & 4 & \text{NIL} & \text{NIL} \\ \text{NIL} & \text{NIL} & \text{NIL} & 5 & \text{NIL} \end{pmatrix}$$

$$D^{(1)} = \begin{pmatrix} 0 & 3 & 8 & \infty & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & \infty & \infty \\ 2 & 5 & -5 & 0 & -2 \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix} \quad \Pi^{(1)} = \begin{pmatrix} \text{NIL} & 1 & 1 & \text{NIL} & 1 \\ \text{NIL} & \text{NIL} & \text{NIL} & 2 & 2 \\ \text{NIL} & 3 & \text{NIL} & \text{NIL} & \text{NIL} \\ 4 & 1 & 4 & \text{NIL} & 1 \\ \text{NIL} & \text{NIL} & \text{NIL} & 5 & \text{NIL} \end{pmatrix}$$

$$D^{(2)} = \begin{pmatrix} 0 & 3 & 8 & 4 & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & 5 & 11 \\ 2 & 5 & -5 & 0 & -2 \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix} \quad \Pi^{(2)} = \begin{pmatrix} \text{NIL} & 1 & 1 & 2 & 1 \\ \text{NIL} & \text{NIL} & \text{NIL} & 2 & 2 \\ \text{NIL} & 3 & \text{NIL} & 2 & 2 \\ 4 & 1 & 4 & \text{NIL} & 1 \\ \text{NIL} & \text{NIL} & \text{NIL} & 5 & \text{NIL} \end{pmatrix}$$

$$D^{(3)} = \begin{pmatrix} 0 & 3 & 8 & 4 & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & 5 & 11 \\ 2 & -1 & -5 & 0 & -2 \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix} \quad \Pi^{(3)} = \begin{pmatrix} \text{NIL} & 1 & 1 & 2 & 1 \\ \text{NIL} & \text{NIL} & \text{NIL} & 2 & 2 \\ \text{NIL} & 3 & \text{NIL} & 2 & 2 \\ 4 & 3 & 4 & \text{NIL} & 1 \\ \text{NIL} & \text{NIL} & \text{NIL} & 5 & \text{NIL} \end{pmatrix}$$

$$D^{(4)} = \begin{pmatrix} 0 & 3 & -1 & 4 & -4 \\ 3 & 0 & -4 & 1 & -1 \\ 7 & 4 & 0 & 5 & 3 \\ 2 & -1 & -5 & 0 & -2 \\ 8 & 5 & 1 & 6 & 0 \end{pmatrix} \quad \Pi^{(4)} = \begin{pmatrix} \text{NIL} & 1 & 4 & 2 & 1 \\ 4 & \text{NIL} & 4 & 2 & 1 \\ 4 & 3 & \text{NIL} & 2 & 1 \\ 4 & 3 & 4 & \text{NIL} & 1 \\ 4 & 3 & 4 & 5 & \text{NIL} \end{pmatrix}$$

$$D^{(5)} = \begin{pmatrix} 0 & 1 & -3 & 2 & -4 \\ 3 & 0 & -4 & 1 & -1 \\ 7 & 4 & 0 & 5 & 3 \\ 2 & -1 & -5 & 0 & -2 \\ 8 & 5 & 1 & 6 & 0 \end{pmatrix} \quad \Pi^{(5)} = \begin{pmatrix} \text{NIL} & 3 & 4 & 5 & 1 \\ 4 & \text{NIL} & 4 & 2 & 1 \\ 4 & 3 & \text{NIL} & 2 & 1 \\ 4 & 3 & 4 & \text{NIL} & 1 \\ 4 & 3 & 4 & 5 & \text{NIL} \end{pmatrix}$$

Topic 20

Review Questions

1. Which of the following **cannot** be present in a graph if we wish to find the shortest path?
   (a) negative weight edges
   (b) cycles
   (c) cycles with negative weight edge
   (d) negative weight cycles
2. Which of the shortest path algorithm is greedy?
3. Which algorithm would you use on a general graph that has negative weight on edges?
4. Prove that $\forall n \in \mathbb{Z}^+$ then
$$\sum_{i=1}^{n}(-1)^i i^2 = \frac{(-1)^n(n)(n+1)}{2}$$
5. Show that if f(n) and g(n) are monotonically decreasing functions then so are the sum, f(g(n)).
6. A graph with n vertices and n-1 eges is either disconnected or a tree.

   (TRUE/FALSE)
7. For every n there exists a directed graph on n vertices with $\Omega(n^2)$ edge that has a topological ordering.

   (TRUE / FALSE)
8. Solve the recurrence T(n) = $4T(\frac{n}{2}) + O(n)$ by finding its running time.
9. Design the most efficient algorithm you can to combine M sorted list, each of size N into one sorted list. give the time complexity of your solution in terms of N and M.
10. Show that the function f(n) = $n^2 + n \rightarrow \Theta(n^2)$
11. Given an array of N non unique values, design the most efficient algorithm you can to find the most common value. Give the time complexity of your solution in terms of N.
12. A minimum weight edge in a connected graph G must belong to <u>every</u> minimum spanning tree for G.

    TRUE/FALSE

13. A minimum weight edge in a connected graph G must belong to <u>some</u> minimum spanning tree for G.

TRUE/FALSE

14. If an edge (u,v) is contained in some minimum spanning tree for graph G then it is a light edge crossing some cut of G.

TRUE/FALSE

15. What problem solving strategy do both Kruskal's and Prim's MST algorithms use?

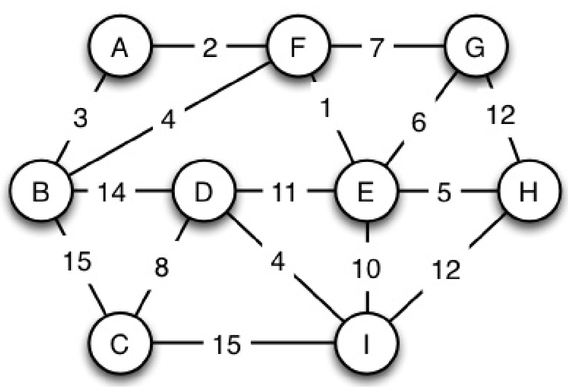
16. Claim: Prim's algorithm correctly finds minimum spanning trees in connected graphs with negative weights.

TRUE/FALSE

17. Claim: Kruskal's algorithm correctly finds minimum spanning trees in connected graphs with negative weights.

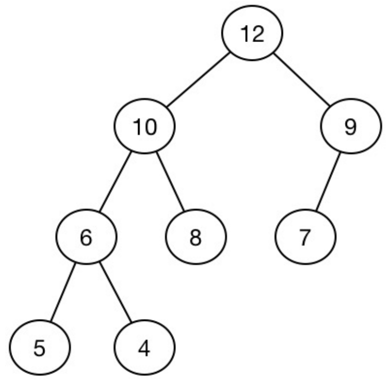
TRUE / FALSE

18. Check off the edges that are included in a Minimum Spanning Tree for this graph.

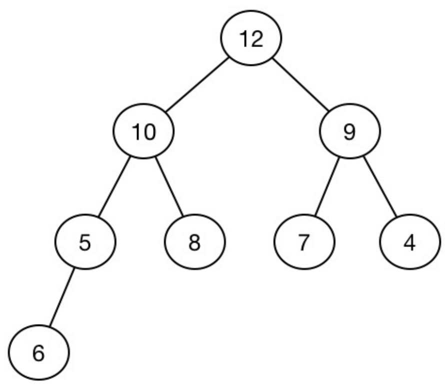


19. In terms of n = $|V|$, Floyd-Warshall's runtime is

20. Is this a Max-Heap? If not, why not?



21. Is this a Max-Heap? If not, why not?



22. Suppose you have a binary heap represented as an array using 1-based indexing (root is at index 1). An item is at index 31 in the array: what is the index of its parent? Write an integer:

23. Suppose you have a binary heap represented as an array using 1-based indexing (root is at index 1). An item is at index 31 in the array: what is the index of its right child? Write an integer:

24. Show the array after Build-Max-Heap has run by entering integers for the keys:

1 2 3 4 5

25. What is the worst-case runtime of randomized quicksort?

26. What is the expected runtime of randomized quicksort?

27. What is the best-case runtime of randomized quicksort?

28. Suppose that you knew that your array was sorted except for the possible misplacement of one or two elements: Which of the sorts that we have studied would be the fastest? What is its expected big-O runtime given your choice in the above question?

29. What is the smallest possible depth of a leaf in a decision tree for a comparison sort of n items?

30. Suppose the deterministic Partition (A, 1, 5) is called on the array A = [7, 3, 9, 4, 5] shown in the table below with 1-based indexing. You will show the state of the array after each swap of elements in the array

31. What is the smallest possible height of a decision tree for a comparison sort of n items?

32. The runtime of counting sort is $\Theta(n + k)$. What is k?

33. The runtime of radix sort using counting sort is $(d(n + k))$. What is d?

34. Which of the sorting algorithms are stable sorts?

35. Use Radix Sort on the following words:

    BOW, DOG, FAX, DIG, BIG, COW

36. Which of the sorting algorithms are not stable sorts?

37. We use Red-Black trees to simultaneously represent what?

38. What of the following properties is violated in a red-black tree when we have underflow of a node in the corresponding 2-4 tree?

39. What of the following properties is necessary to guarantee that the 2-4 tree it represents is balanced, assuming that the other properties hold?

40. What of the following properties is violated in a red-black tree when we have either incorrect representation or overflow of a node in the corresponding 2-4 tree?