University of Colorado
Boulder

CSCI 3104 Algorithms

Fall 2015
Lecture 28 (Oct 30)

Announcements (1)

- Midterm Exam 2
  - Wed Nov 4
  - in class, 45 minutes, arrive on time
  - closed-book
  - materials covered since midterm exam 1

Announcements (2)

- Office hours
  - Wed 2-3pm, ECCR 1B05C (Qin Lv)
  - Mon 2-3pm, ECCS 112A (Wanshan Yang)
  - Mon 3-4pm, ECCR 1B06 (William Mortl)
  - Tue 11am-12pm, ECCS 122, (Shuo Zhang)
  - PLA https://foundation.cs.colorado.edu/la/
- William Mortl's Office Hour This Week
  - Fri Oct 30, 3-4pm, ECCR 1B06

Problem 1(a)

- Determine if the following statements are TRUE or FALSE. Briefly explain why.
  - (a) Given the same graph and the same starting vertex, DFS (depth-first-search) and BFS (breadth-first-search) always visit the vertices in the same order.
  - FALSE. DFS is stack based, follow one branch to bottom; BFS is queue based, check all neighbors first, layer-by-layer

Problem 1(b)

- Determine if the following statements are TRUE or FALSE. Briefly explain why.
  - (b) No greedy algorithm can guarantee optimal solution.
  - FALSE. E.g., fractional knapsack, Kruskal's or Prim's algorithm for MST, etc.

Problem 1(c)

- Determine if the following statements are TRUE or FALSE. Briefly explain why.
  - (c) Dijkstra's algorithm cannot always find the shortest path if the graph contains negative edges.
  - TRUE. Dijkstra's algorithm assumes positive edges and growing path lengths, doesn't re-examine shortest paths to vertices that were popped out of queue

Problem 1(d)

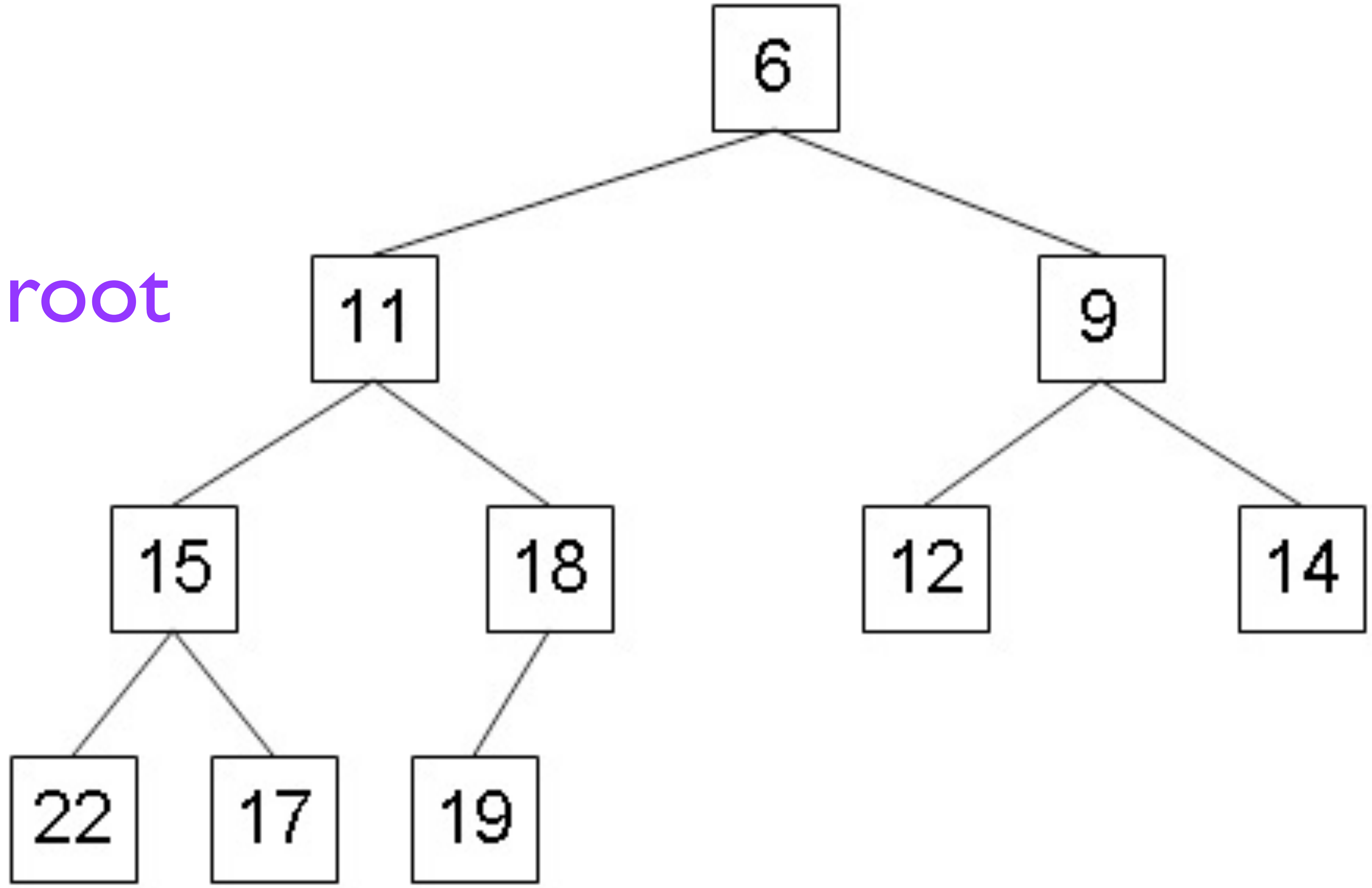
- Determine if the following statements are TRUE or FALSE. Briefly explain why.
  - (d) Given the same graph, Kruskal's algorithm and Prim's algorithm do not always find the same minimum spanning tree.
  - TRUE. Kruskal's selects edges with increasing length; Prim's selects shortest edge between selected vertices and unselected vertices

Problem 2(a)

- Show the key steps of the following computations:
  - (a) Huffman coding of 7 symbols with the following frequencies:
  - {A : 16, B : 3, C : 17, D : 25, E : 8, F : 19, G : 12}
  - B:3 + E:8 => BE: 11 + G:12 ==> BEG: 23
  - A:16 + C:17 => AC: 33
  - F:19 + BEG:23 => BEFG: 42
  - D:25 + AC:33 => ACD: 58 + BEFG: 42 => 100

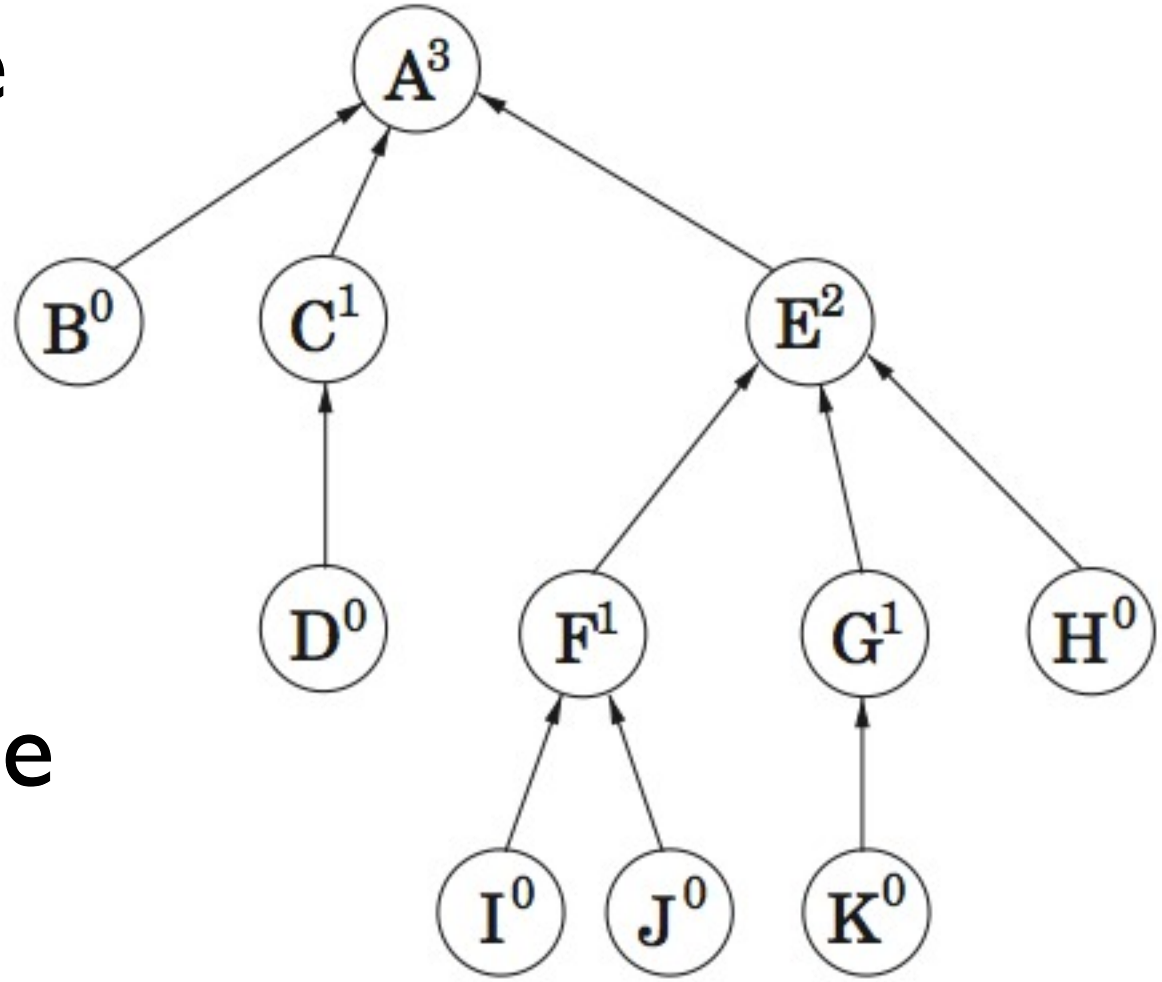
Problem 2 (b)

- (b) Given the binary heap below, show the resulting binary heap after a delete_min() operation.
  - remove 6
  - move 19 to root
  - 19 <==> 9
  - 19 <==> 12



Problem 2(c)

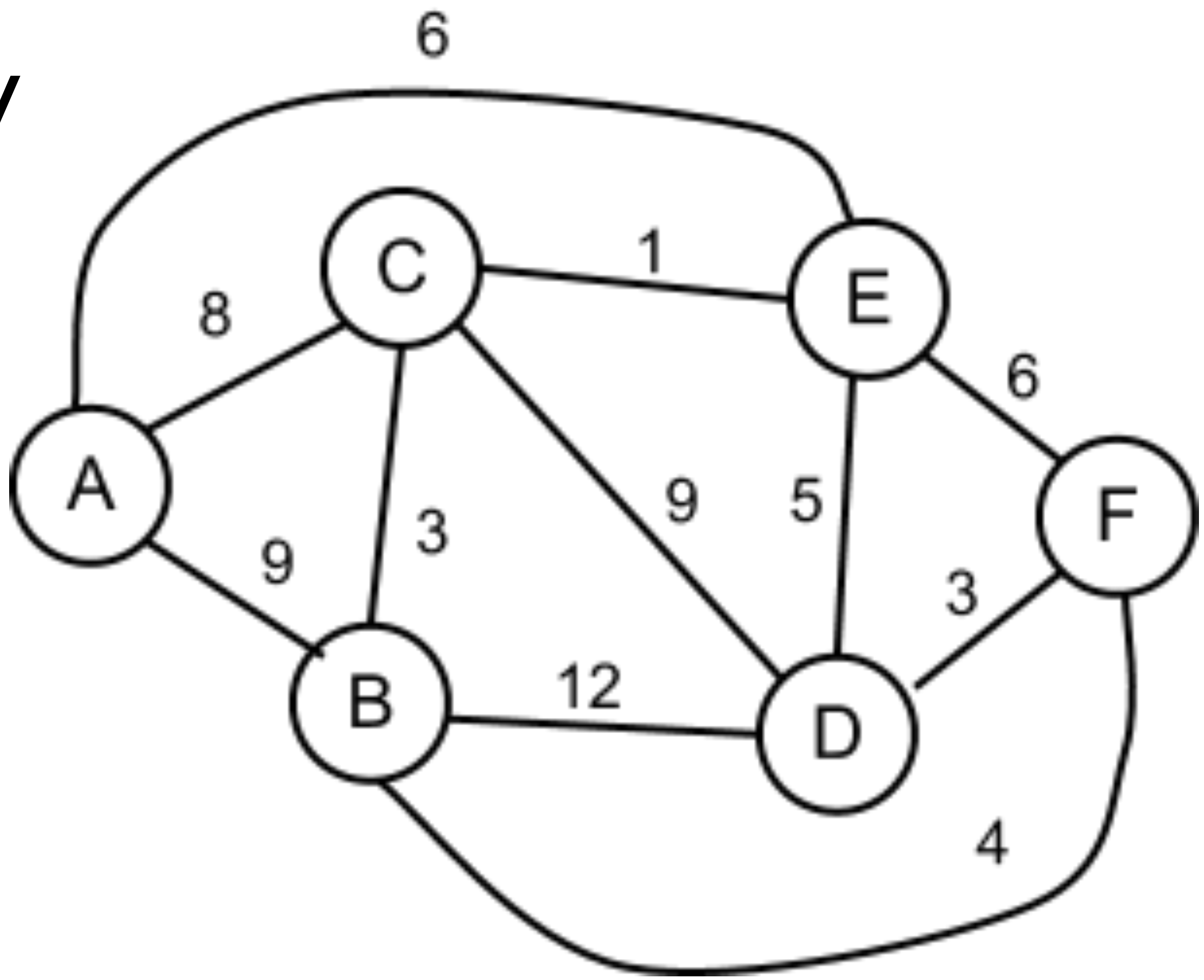
- Show the key steps of the following computations:
    - (c) When using path compression in the find(J) operation, what changes are made to the directed tree shown below?
    - J ==> A, F ==> A



Problem 3(a)

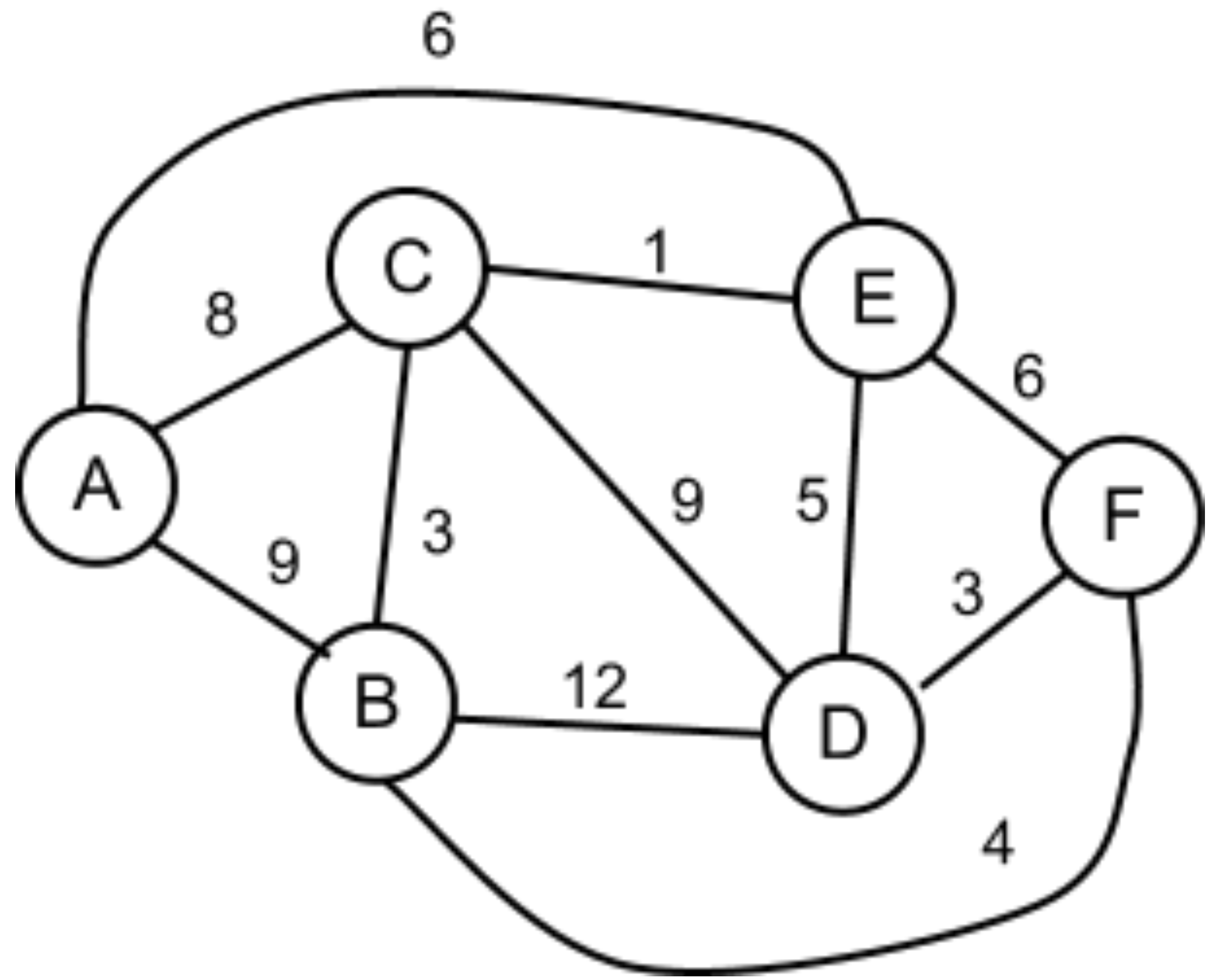
- Given the graph below
  - (a) When using Kruskal's algorithm to find the minimum spanning tree of the graph, which edges are selected, in what order?

    C--E, C--B, D--F, B--F, A--E



Problem 3(b)

- Given the graph below
  - (b) When using Dijkstra's algorithm to find the shortest paths from A, in which order are the vertices examined (via delete-min())? What is the shortest distance from A to each vertex?



Problem 3(b): Dijkstra's Algr.

	A	B	C	D	E	F
	0	∞	∞	∞	∞	∞
A	0	9	8	∞	6	∞
E	0	9	7	11	6	12
C	0	9	7	11	6	12
B	0	9	7	11	6	12
D	0	9	7	11	6	12
F	0	9	7	11	6	12

Problem 4

- Interleaving strings. Given three strings $A = (a_1, a_2, \ldots, a_n)$, $B = (b_1, b_2, \ldots, b_m)$, and $C = (c_1, c_2, \ldots, c_{n+m})$, determine if C can be obtained by interleaving all characters in A and B while preserving the ordering of these characters in the original A and B strings. For example, AmSKe can be obtained by interleaving ASK and me. Design a dynamic programming algorithm to solve this problem in $O(nm)$ time.

Problem 4

- $S(i, j)$: Is $(c_1, c_2, \ldots, c_{i+j})$ an interleaving string of $(a_1, a_2, \ldots, a_i)$ and $(b_1, b_2, \ldots, b_j)$? True or False
- Four cases when comparing c_{i+j}, a_i and b_j
  - $c_{i+j} \mathrel{!=} a_i \;\&\&\; c_{i+j} \mathrel{!=} b_j$ False
  - $c_{i+j} == a_i \;\&\&\; c_{i+j} \mathrel{!=} b_j$ $S(i-1, j)$
  - $c_{i+j} \mathrel{!=} a_i \;\&\&\; c_{i+j} == b_j$ $S(i, j-1)$
  - $c_{i+j} == a_i \;\&\&\; c_{i+j} == b_j$ $S(i-1, j)$ or $S(i, j-1)$
  - e.g., A = aS, B = bS, C = aSbS