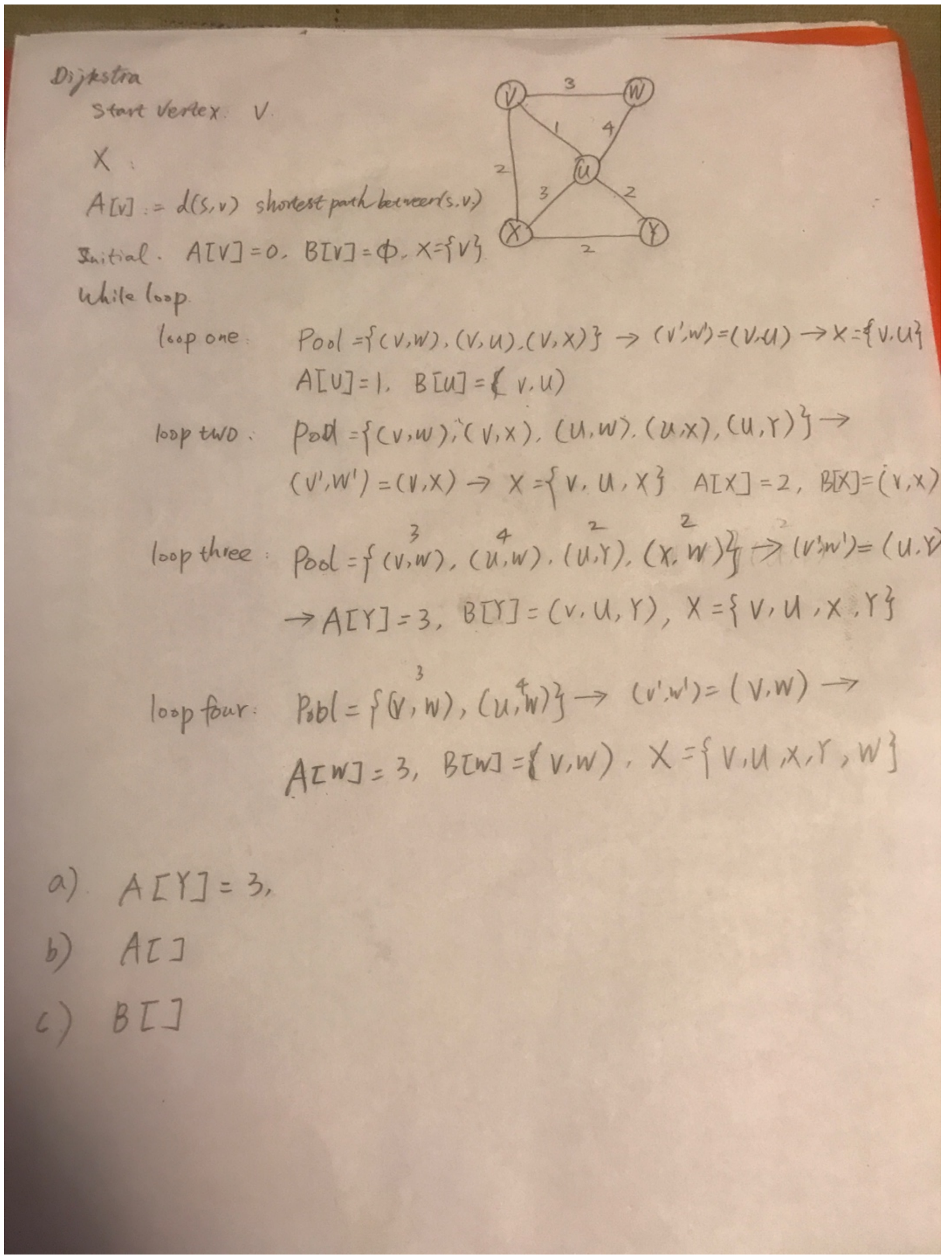
Yuliang Jin 986381

Problem 1

Must every dense graph be connected? Explain. No, A dense graph is a graph with theta(n^2) edges, it's not necessarily a connected graph.

Problem 2



Dijkstra

Start Vertex: V.

X :

$A[v] := d(s,v)$ shortest path between (s,v)

Initial. $A[V]=0$, $B[V]=\Phi$, $X=\{V\}$

while loop.

loop one: Pool $=\{(V,W),(V,U),(V,X)\} \rightarrow (v',w')=(V,U) \rightarrow X=\{V,U\}$

$A[U]=1$, $B[U]=\{(V,U)$

loop two: Pool $=\{(V,W),(V,X),(U,W),(U,X),(U,Y)\} \rightarrow$

$(V',W')=(V,X) \rightarrow X=\{V,U,X\}$ $A[X]=2$, $B[X]=(V,X)$

loop three: Pool $=\{\overset{3}{(V,W)},\overset{4}{(U,W)},\overset{2}{(U,Y)},\overset{2}{(X,W)}\} \rightarrow (v',w')=(U,Y$

$\rightarrow A[Y]=3$, $B[Y]=(V,U,Y)$, $X=\{V,U,X,Y\}$

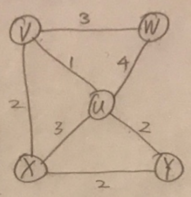
loop four: Pool $=\{\overset{3}{(V,W)},\overset{4}{(U,W)}\} \rightarrow (v',w')=(V,W) \rightarrow$

$A[W]=3$, $B[W]=\{(V,W)$. $X=\{V,U,X,Y,W\}$

a) $A[Y]=3$,

b) $A[\,]$

c) $B[\,]$



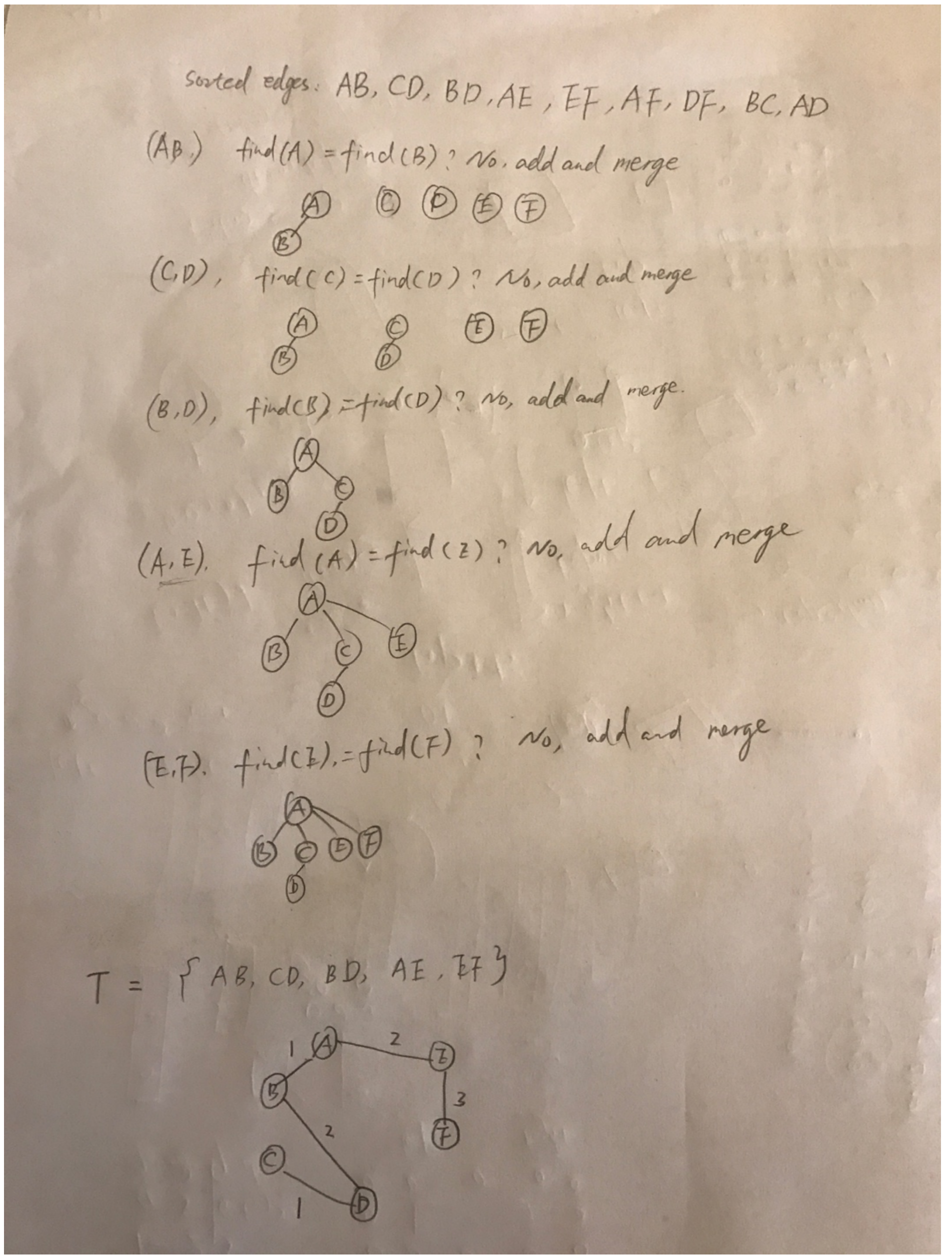
Problem 3

a. There is no shortest path. b. The Dijkstra's approach is better. Cause BFS search become too high when the weight between becomes bigger(say 10000, the height would be at lest 10000)

Problem 4

Describe an algorithm for deleting a key from a heap-based priority queue that runs in O(log n) time, where n is the number of nodes. (Hint: You may use auxiliary storage as the priority queue is built and maintained. Assume there are no two nodes have the same key.) This technique is needed for the optimized Dijkstra algorithm discussed in the slides. a. Store the heap using an array b. Add a hash table the key is the value of the vertex, and the value is the index of the vertex in the array of the heap. c. Whenever add a vertex, remove the root, down heap operation happens, update the hash table to make sure any time the hash table store the accurate values d. When to delete a vertex, locate it takes O(1) and rebalence operation takes O(logn), so the deletion also takes O(logn)

Problem 5



Sorted edges: AB, CD, BD, AE, EF, AF, DF, BC, AD

(AB) find(A) = find(B)? No, add and merge

(C,D), find(C) = find(D)? No, add and merge

(B,D), find(B) = find(D)? No, add and merge.

(A,E). find(A) = find(E)? No, add and merge

(E,F). find(E), = find(F)? No, add and merge

T = {AB, CD, BD, AE, EF}

Problem 6

The goal of this exercise is to devise a feasible algorithm that decides whether an input integer is prime.

Because the right output of this algorithm each time is 1/2, so in the for loop execute the algorithm a few times the probability of success would be increased.