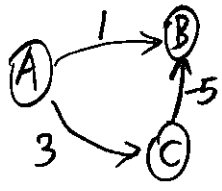
CSE 373 Homework 5
Student number 1568037
Name Changming Feng

1) Negative Graphs (20 points)

(a) [5 Points] If there is more than one minimum cost path from v to w, will Dijkstra's algorithm always find the path with the fewest edges? If not, explain in a few sentences how to modify Dijkstra's algorithm so that if there is more than one minimum path from v to w, a path with the fewest edges is chosen. Assume no negative weight edges or negative weight cycles.

NO.
Create a new map EdgeNo and for each vertex w, EdgeNo.get(w) returns the minimum number of edges from source v to w. Each time marking the unknown nodes v with lowest cost as known, and if there are more than one nodes has the same lowest cost, choose the one with smallest value in the map, ie. EdgeNo.get(v). v is the known vertex, w is adjacent to v. Update the path and EdgeNo.get(w) if EdgeNo.get(v)+1 < EdgeNo.get(w) after updating the weight of w. when necessary.

(b) [5 Points] Give an example where Dijkstra's algorithm gives the wrong answer in the presence of a negative cost edge but no negative-cost cycles. Explain briefly why Dijkstra's algorithm fails on your example. The example need not be complex; it is possible to demonstrate the point using as few as 3 vertices.

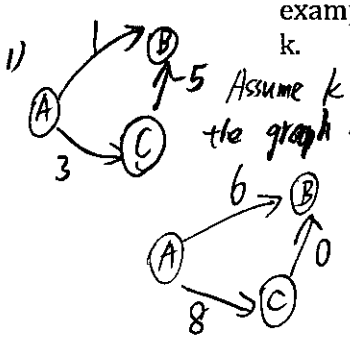


Dijkstra's algorithm will fail to find the shortest path from A to B (A->C->B). Since at first step from A, the cost of B is 1, cost of C is 3, the vertex B will be chosen and marked as known. Once the B is known with cost of 1, it means the shortest path to B has been found. Thus it will not update the path and cost of B. -)

(c) [10 Points] Suppose you are given a graph that has negative-cost edges but no negative-cost cycles. Consider the following strategy to find shortest paths in this graph: uniformly add a constant k to the cost of every edge, so that all costs become non-negative, then run Dijkstra's algorithm and return that result with the edge costs reverted back to their original values (i.e., with k subtracted).

- Give an example where this technique fails (Dijkstra's would not find what is actually the shortest path) and explain why it fails.

- Give a general explanation as to why this technique does not work. Think about your example and why the original least cost path is no longer the least cost path after adding k.

1)


Assume k=5, the graph becomes:

The technique will find A->B with cost of 6, then by substracting k, the result is A->B with cost of 1. It fails to find the path: A->C->B.

Why it fails: from A, the cost of B is 6, less than the cost of C which is 8. Thus B is marked as known with cost of 6. Once B is marked as known, the path A->C->B will not be found.

2) See on top of next page.

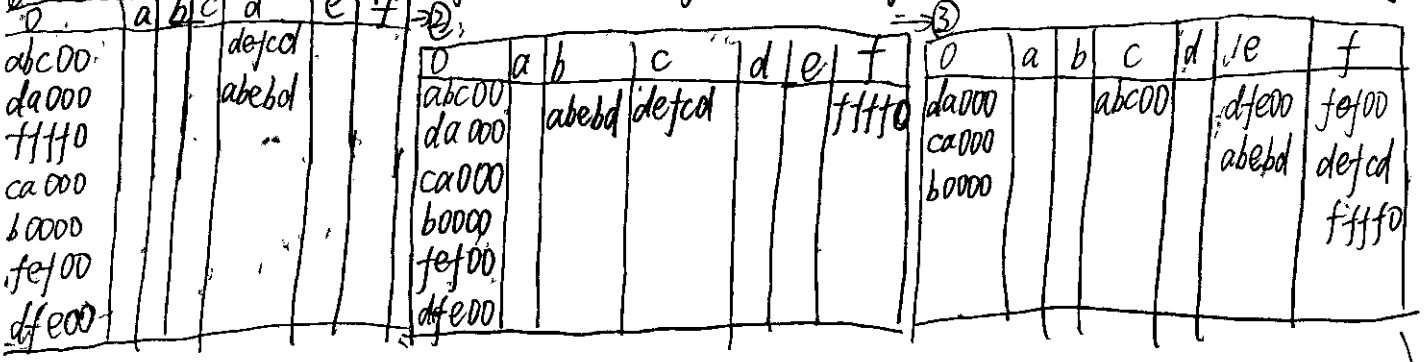
1).c). 2). This technique doesn't work because different paths contain different amount of edges, which means different amount of k will be added to different paths. Since a constant k is added to each edge, the original least cost path(path1) contains m edges, which means total (mxk) will be added to the whole path. Some other path(path2) contains n edges. When $n < m$, less k will be added to path2. Thus there is a chance the cost of path2 is less than that of path1 after adding k.

2) Sorting (10 points)

1) Using RadixSort with a radix of 6 (letters a,b,c,d,e,f) to alphabetically sort the following strings, draw contents of each bucket at the end of each iteration.

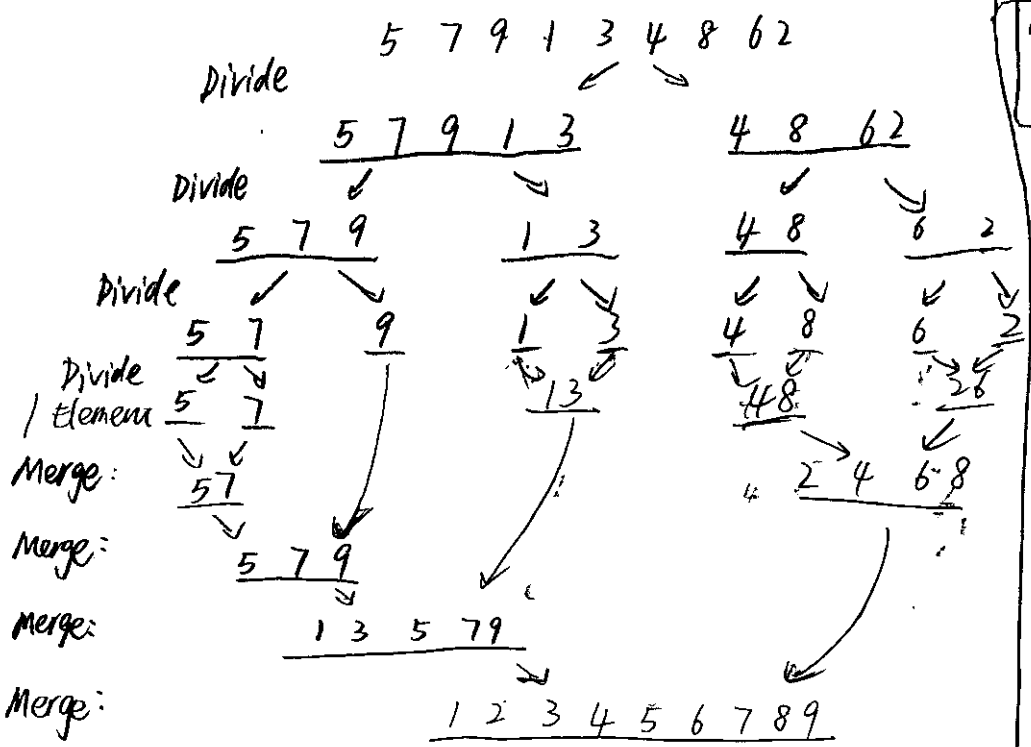
`Strings = (abc, da, ffff, defcd, abebd, ca, b, fef, dfe)`

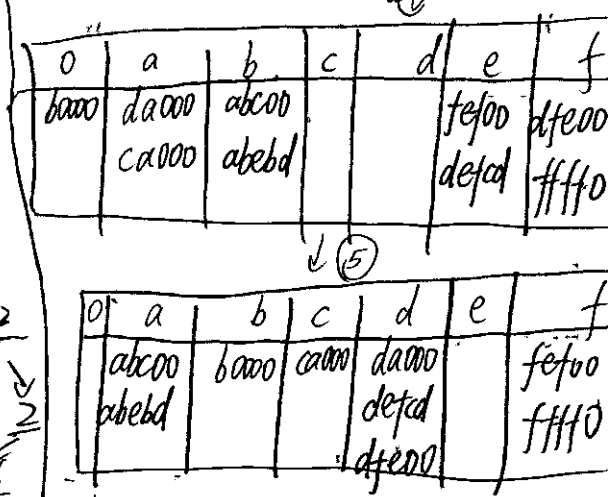
Add '0's to the end of string to make all strings the same length, and assume 0 is smaller than any letter.



2) Using MergeSort, sort the following list of numbers. Show your work by drawing the merge-tree (as seen in the lecture slides) to show each merge and join of intermediate steps. Assume that when there are odd number of elements, that the left 'half' will include the extra element.

`Numbers = [5,7,9,1,3,4,8,6,2]`





Remove 0's. get the result:

abc, abebd, b, ca, da, defcd, dfe, fef, ffff.