Jeonghyeon Woo

CS415

HW2

1.

a. Block neseted join is scanning S once for every tuple in R.

   Cost: $\mathrm{N}_{\text{block of S}} + \frac{\mathrm{N}_{\text{block of S}}}{buffer-2} * N_{blockofR}$ = 4200

b. Sort-merge algorithm is sorting both relations and merging them, and then scan each tuples.

   Cost: $2\mathrm{b}_\mathrm{s}\left(1 + \log_{\mathrm{B}-1}\left(\frac{\mathrm{b}_\mathrm{S}}{\mathrm{B}-2}\right)\right) + 2\mathrm{b}_\mathrm{r}(1 + \log_{\mathrm{B}-1}\left(\frac{b_r}{B-2}\right) + b_S + b_r$ = 6000

c. Hash join is parting and matching the tuples of relations.

   Cost: $3(\mathrm{b}_\mathrm{s} + b_r) = 3600$

2.

a. Use DFS algorithm.

```
DFS(G, Q)
v=s //current point is source point
Label1:
    Loop1:
        If v->e.label ∉ M then break
        If v==t then return 1
        Else if v->child is null then break (=goto Loop2)
        Else v=v->child and goto Loop1
    //search until the deepest
    Loop2:
        If v==s then return 0
        else if v->sibling is not null then v=v->sibling and goto label1
        else v=v->parent and goto Loop2
    //do backtrack if current point reaches the deepest point
```

b. DFS(G, u, v)

mostreliable=null  //it is returning edge

Label1:

Loop1:

If v->e=(u,v)

If r(v->e)==1 return v

Else mostreliable=v

If v->child is null then break (=goto Loop2)

Else v=v->child and goto Loop1

//search until the deepest

Loop2:

If v->parent is null then return mostreliable

Else if v->sibling is not null then v=v->sibling and goto Label1

Else v=v->parent and goto Loop2

//do backtrack if current point reaches the deepest point

//Finds the first edge of u,v with r(edge) is 1, or the last edge of u,v if no edge have r(edge) value of 1.

Correctness proof:

Assume: input = u, v output = v

Proof:

Suppose matching of input u,v to edges in graph G is done without an error.

Searching until the deepest point is successful since matching is done without an error

Backtracking is successful since searching is done successfully.

While searching, the loop will be end when r(edge) is 1 since matching and searching are done successfully.

The loops of searching and backtracking will be successfully end since matching is done without an error.

In the iteration of program, the loops will be end safely while matching is done without an error, and it is shown that program will return the first matching path with r(edge) value of 1 or last matching path with r(edge) value of 0. Since any edge with r(edge) value of 1 is the most reliable path, the program gives the most reliable path when it returns

those edges. Since any edge with r(edge) value of 0 is the most reliable path unless there exists edge with r(edge) value of 1, the program gives the most reliable path when it returns 0.

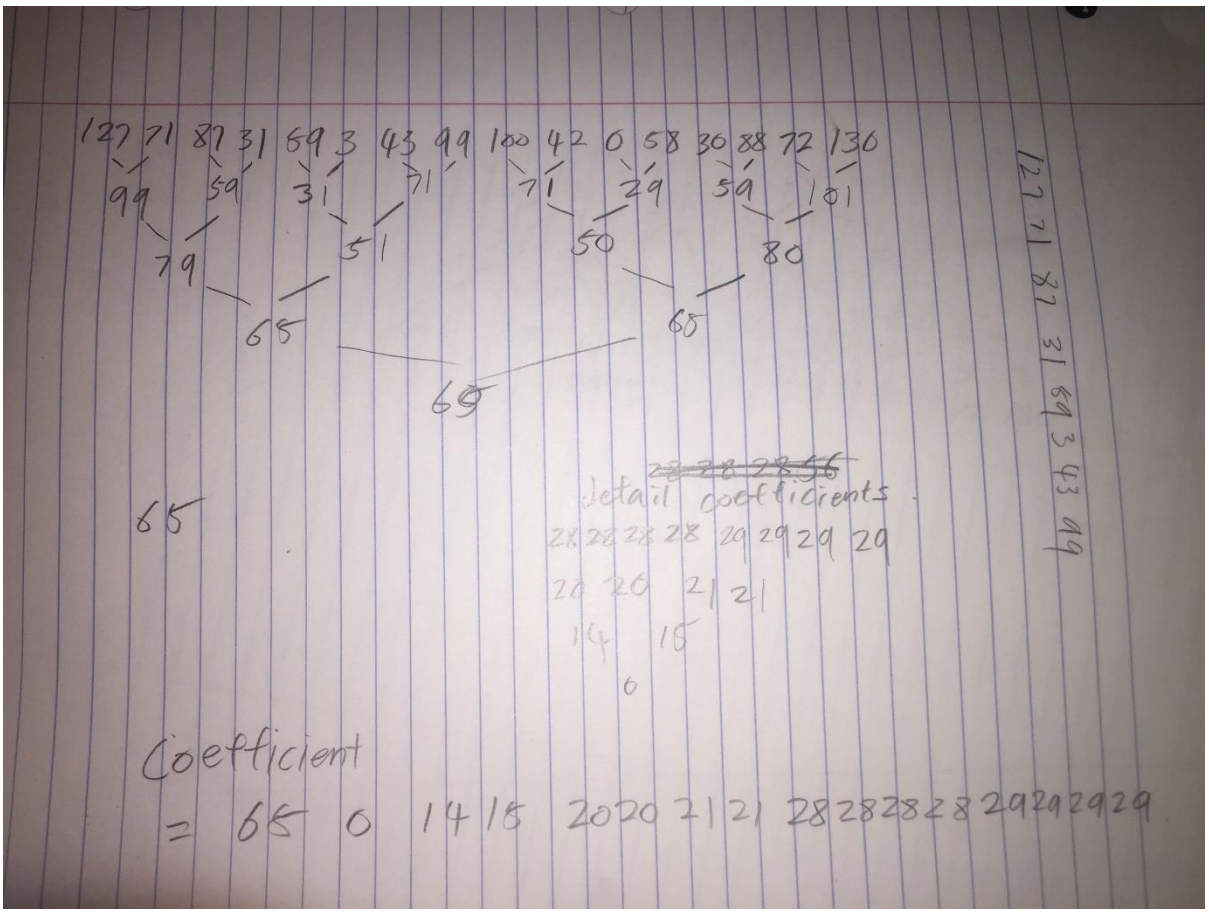
Therefore, the program gives the most reliable path.

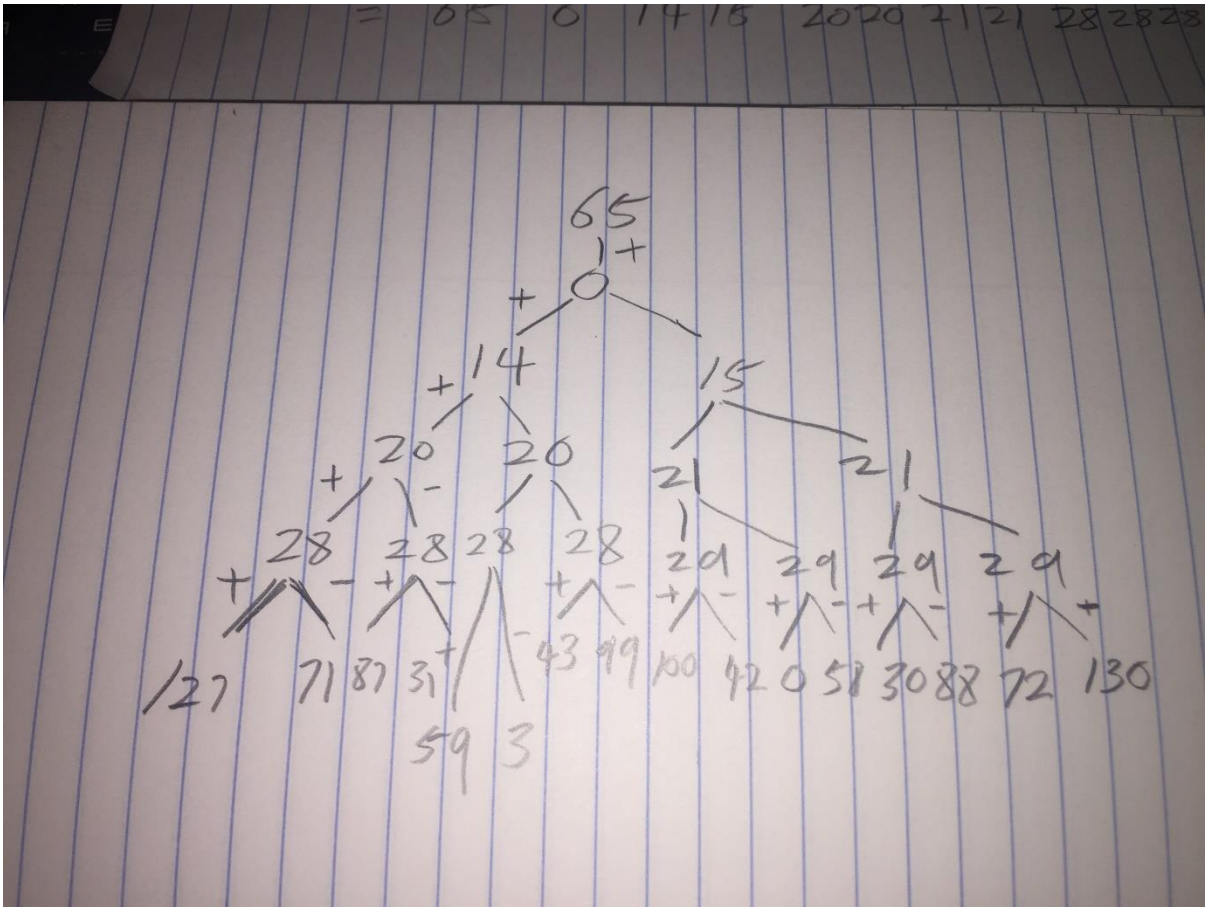
Complexity is O(|V|+|E|) since it is depth first search.

c. 2 hop cover can give paths between source and target. For the algorithm of part a, 2 hop cover can be used to find paths between s and t, so the program only needs to check if the edges in the paths have label in M. For example, $L_{out}(s) \cap L_{in}(t)$ can show the paths. For the algorithm of part b, 2 hop cover can be used to find edges that contain u and v, so the program only needs to find edges with value r(edge) of 1.

3.

a.






b. 65+14-20-28=31

c. 31+59+3=93