CSE 541 HW4

Pro1.

From the theorem, we know that the theorem means that for integers m and n, each combination of subset of S (m-subset) should have equally chance of showing up, which should be ***Probability (each subset showing up) = 1/().***

Induction:

1, Base case:

For m=0, the algorithm is page 130 will return empty set φ, the theorem is true as there is only one size m subset of [n].

2, Inductive Hypothesis: maintenance:

Assume m>0, and assume the invariant holds for m-1 which means “For any m’ < m, n’ < n, random-sample (m’, n’ ) returns a random m’ -subset of {1, 2, . . . , n’} in which each m’ -subset is equally likely”.

When we pass m, the recursive call returns an m-1 sample with uniform probability, which is 1/(). And then there two choices – either the new m-subset includes n or not.

If n ∈ S, with probability (m/n),

Since Prob(n ∈ S) = prob(i ∉S &&i==n)+ prob(i∈S’) =(m-1/n)+(n-m+1/n)\*( 1/n-m+1) =(m-1/n)+(1/n)= m/n

the probability for each combination including n is:

(m/n)\*(1/() )= ()

if n ∉S, with probability (n-m/n), which is 1-prob(n ∈ S), so it includes one of (n-m) numbers not present. The chance for each is:

(n-m/n)\*(1/() )= ()

Prob2,

FASTER-ALL-PAIRS-SHORTEST-PATHS(W):

n = W.rows

L.1= W

m = 1

while m<n -1:

let L.2m be a new n X n matrix

L.2m = EXTEND-SHORTEST-PATHS(L.m,L.m)

m = 2m

Lcheck = EXTEND-SHORTEST-PATHS(L.m,L.m)

For i = 1 to n

(if Lcheck ij != Lm ij) return exist//correct

If Lcheckii <0:

Print: “the graph contains a negative-weight cycle”

return L.m

We can detect the existence of negative cycles simply by checking whether there are negative values on the diagonal (which is Lii) of the matrix L(n). Since the shortest weight circle might need to n edges, so we should calculate the matrix up to a least Ln which is Lcheck.

Prob3.

Dijkstra calculate the shortest path for two vertexes, but we want to find the max-probability path for two vertexes. Where x is the start vertex, and y is the end vertex. G is the graph, r is probability set.

Pseudo-code:

Relax(u,v,r)

If v.p <u.p\*r(u,v):

v.p = u.p\*r(u,v)

v.π = u

Initialize\_single\_source(G,x):

For each vertex v ∈G.V:

v.p = 0

v. π = Null

x.p =1

Reliability (G, r, x, y)

1. Initialize\_single\_source(G,x)
2. S = Φ
3. Q = G.V
4. While Q != Φ do:
5. U = Extract-Max(Q) // v time
6. S = S U{u}
7. for each vertex v ∈ G.Adj[u] do: // v time
8. Relax(u,v,r)
9. end for
10. end while
11. answer = list<vertex>
12. answer.push\_front(y)
13. while y != x:
14. answer.push\_front(y. π)
15. y = y. π
16. end while
17. answer.push\_front(x)
18. return answer