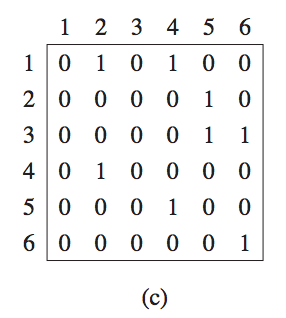
**G(V,E)**: v: 端点, e: 路径

Adj-list-represent: consists of an array of each vertex, like Adj[u] = all the vertices v such that there is an edge(u,v) in E. for directed graph, the sum of length of all the adj list is |E|, for undirected graph, is |2E|. the amount of memory is Θ(E+V).

Another: adj-matrix-representation:

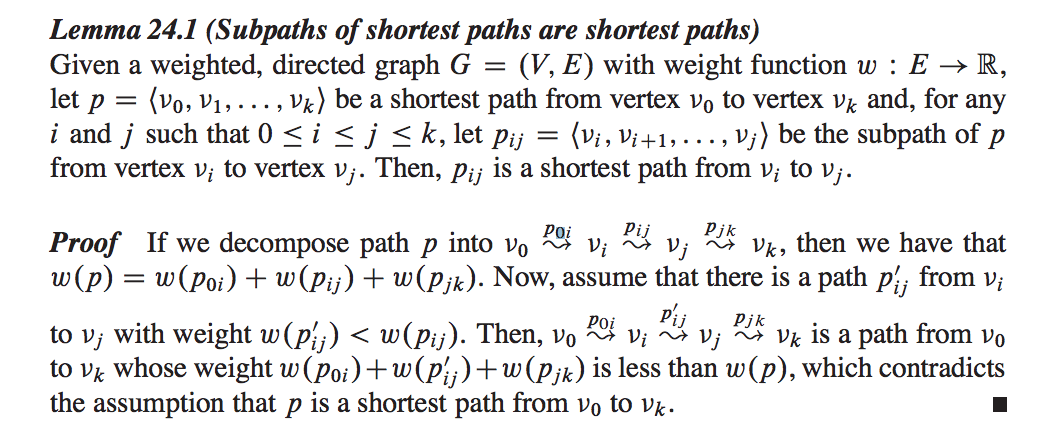
aij = 1 if (i,j) in E, 0 otherwise



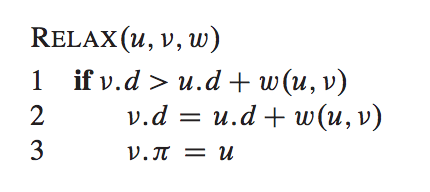
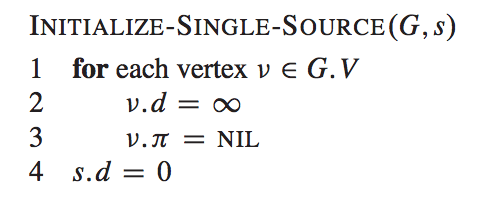
The Bellman Ford algorithm:

Dynamic programing base-

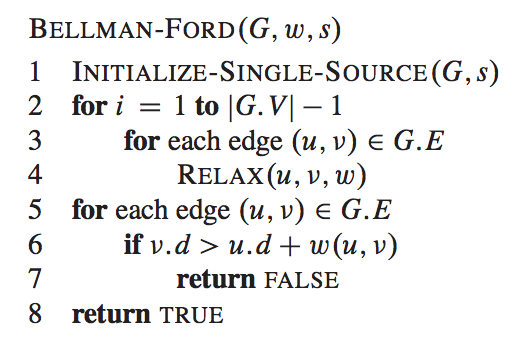
Subgame-property:



initialize: relax:



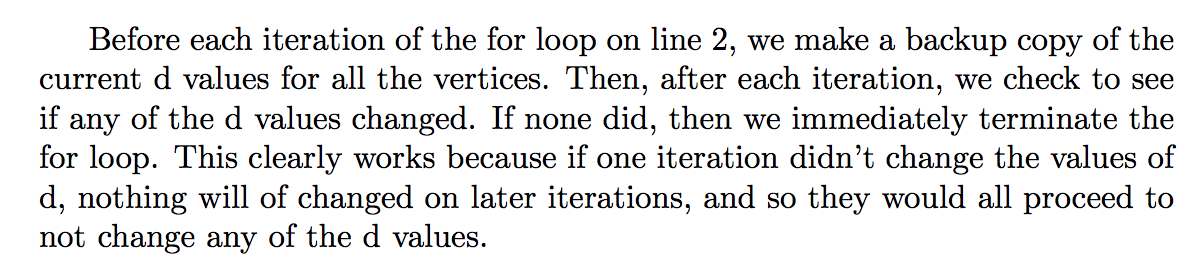
包含negative-edge 没事, but can’t include negative-circle.可以test, nega-circle



2-4 update the s-p, 5-8 check negative circle.

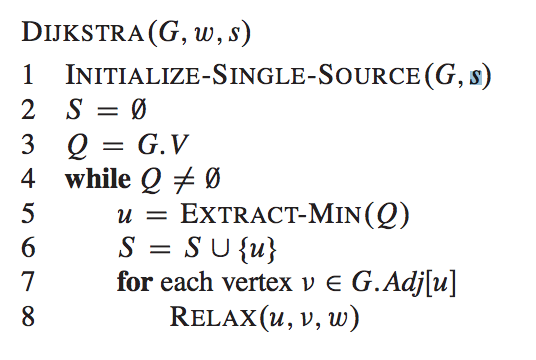
The Bellman-Ford algorithm runs in time O(VE),

Check how many loop:

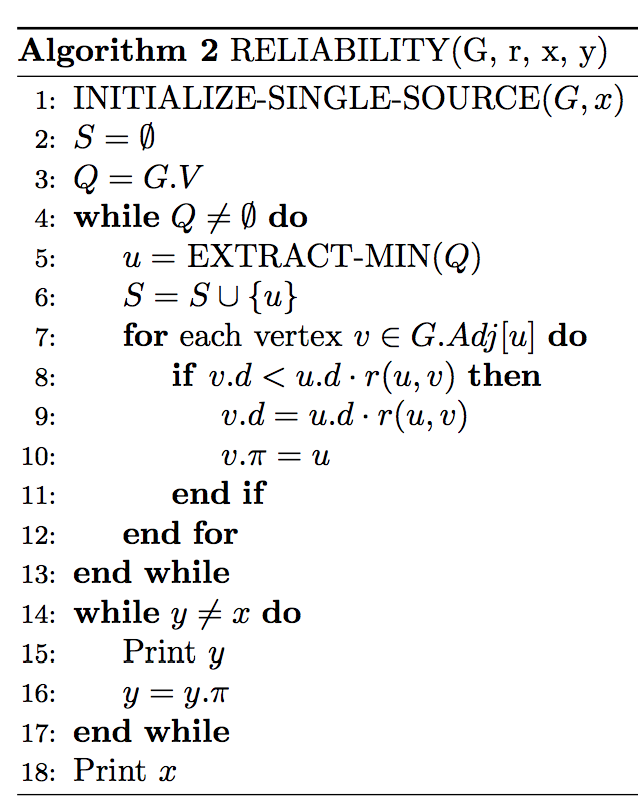


Dijkstra’s shortest path algorithm:

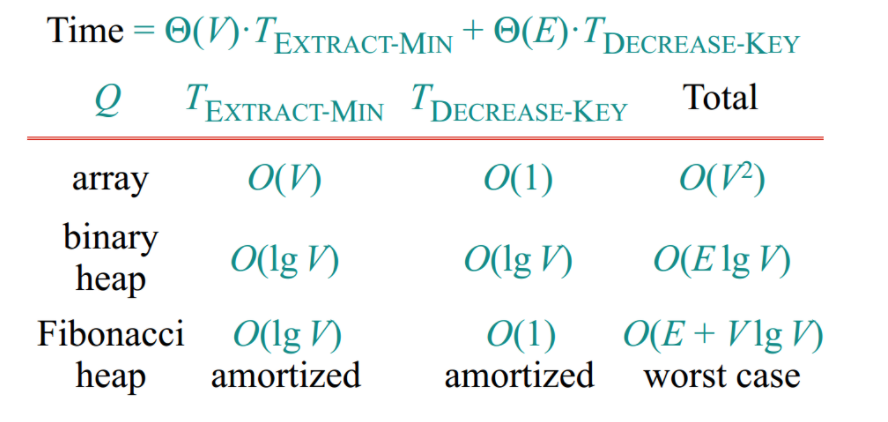
running time: V2 but can be O(ElgV) if we implement the binary min-heap.



求一条通道not fall 的概率



All-pairs-shortest-path:

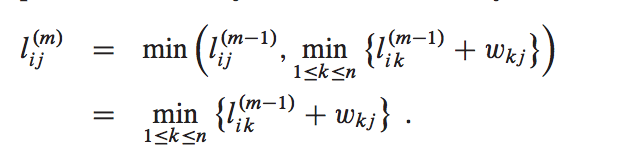


For Dijkstra, with linear-array, time is O(V3), if with binary-min-heap, the time is O(VE\*logV).

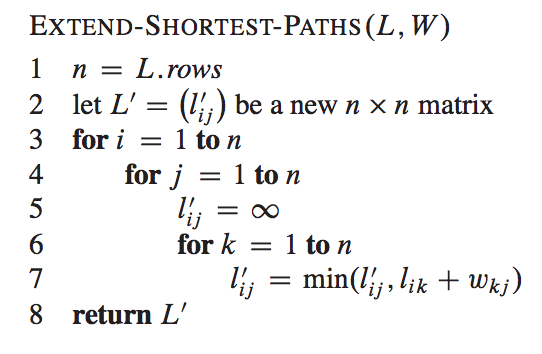
If contains negative edge, we only can use bellman-ford:

The time is O(V2E) = O(V4) for a dense graph, since E= V2.

Sub-game:



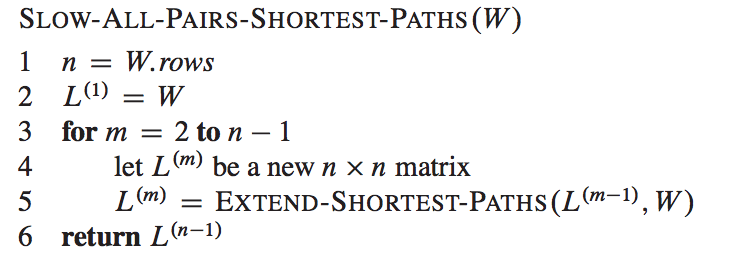
bottom-up: Given a W and L(m-1), return the L(m)

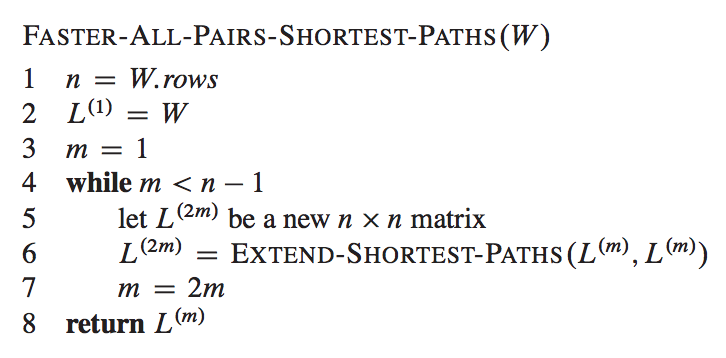


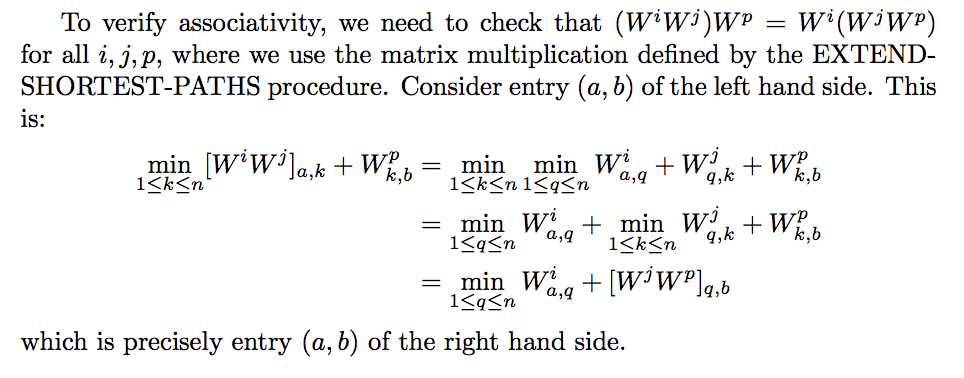
这个算法的实际相当于matrix multiply.

So using extend-method, to calculate the cost:

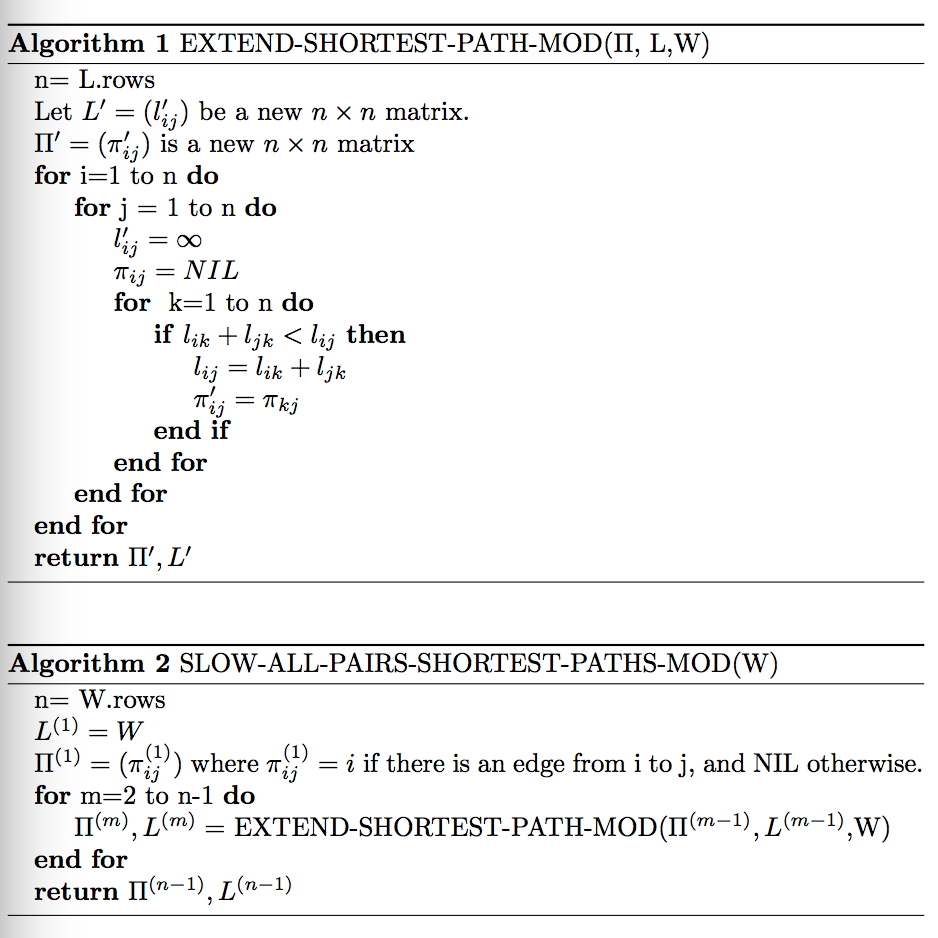
O（n4）

we only need to calculate L(n-1), and instead of adding by 1, we can multiple by 2.

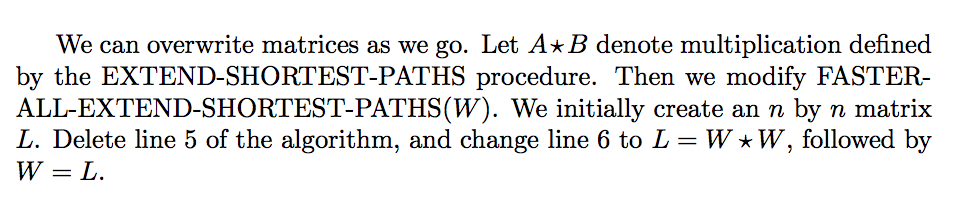
  
Because each of the lg(n-1) matrix products takes n3 time, FASTERALL-PAIRS-SHORTEST-PATHS runs in (n3lg n) time.



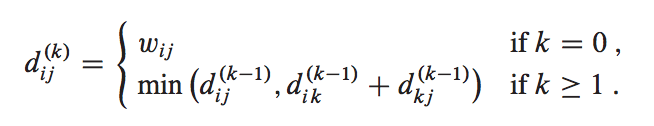
computer the vertices on the S-Paths:

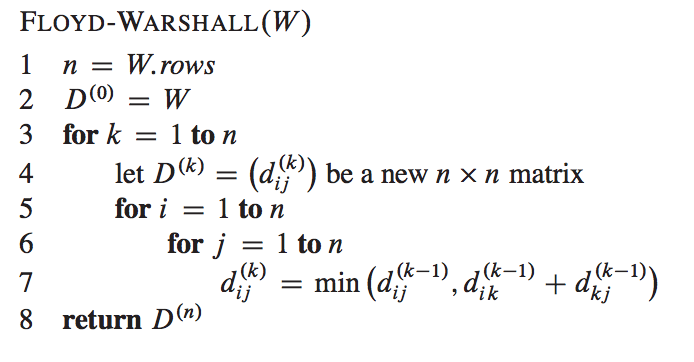


when the memory require is O(n2):



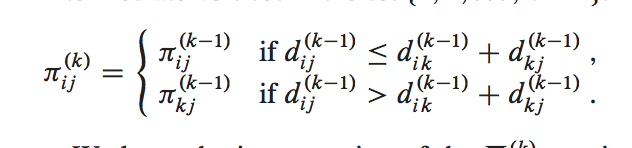
Floyd-Warshall-all pairs-sp:

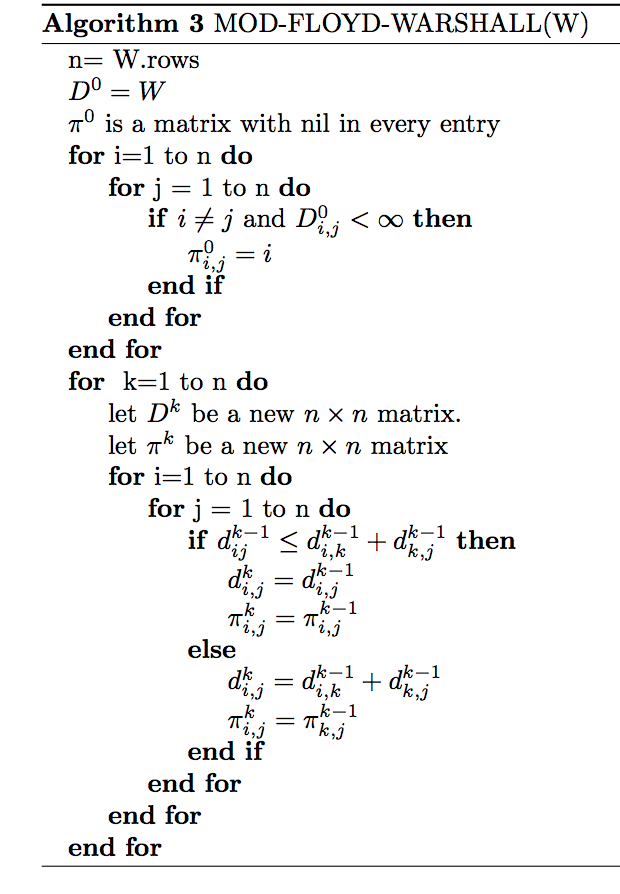




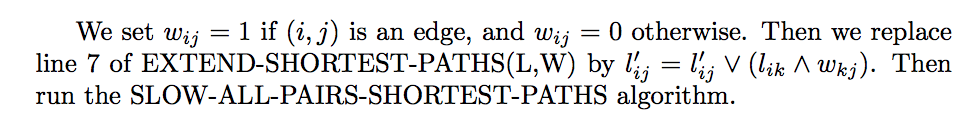
calculate the predecessor matrix from the completed matrix L is O(n3) time.

constructing a shortest path:

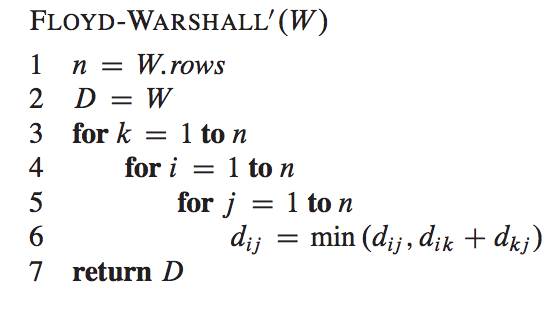


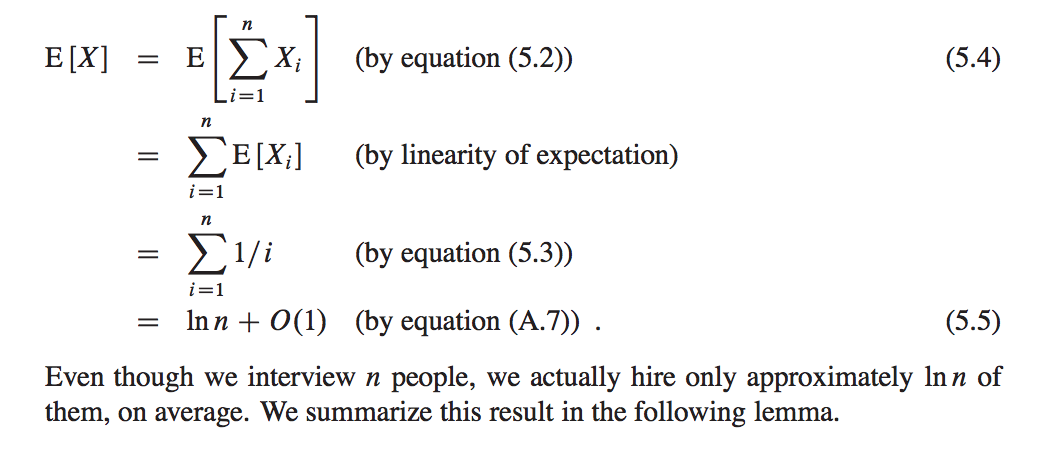


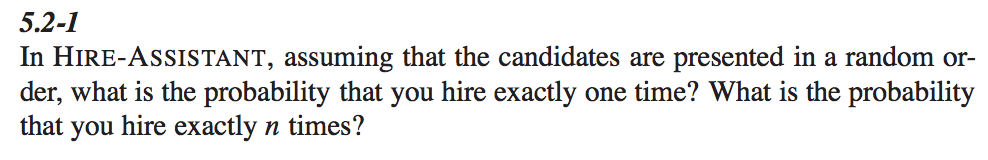
determine whether there is a path from a to b:

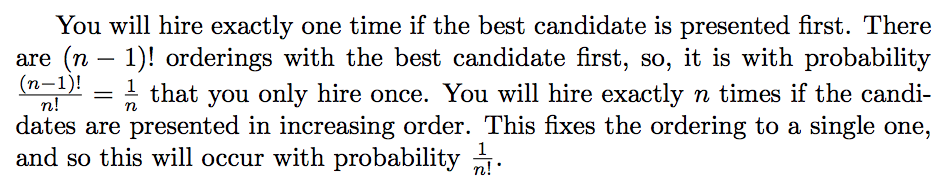


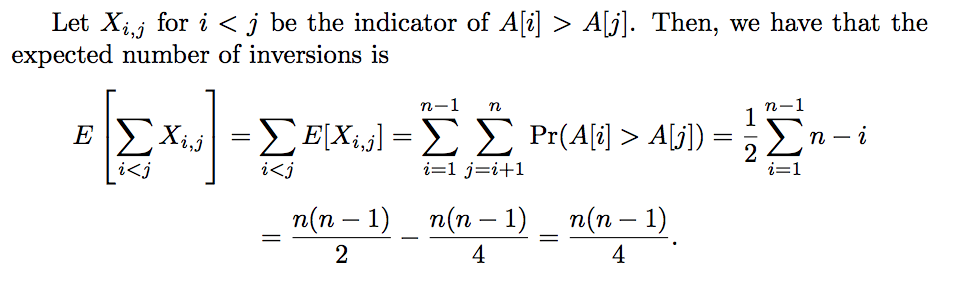
with space O(n2) required of F-W:



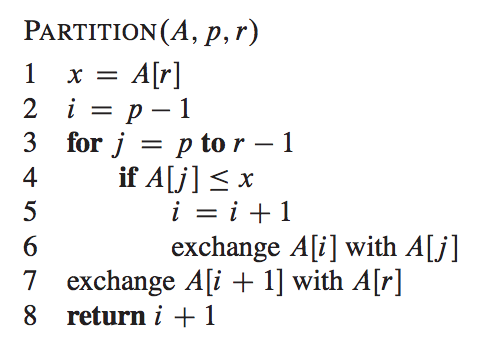


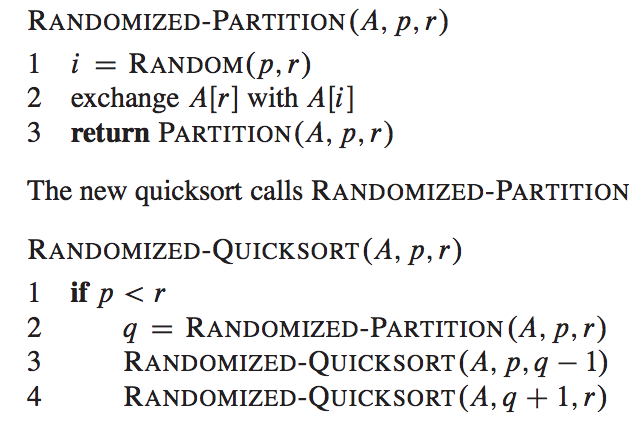




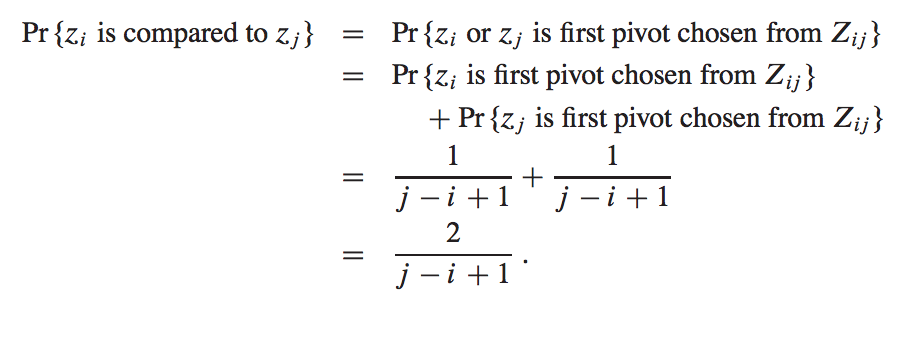


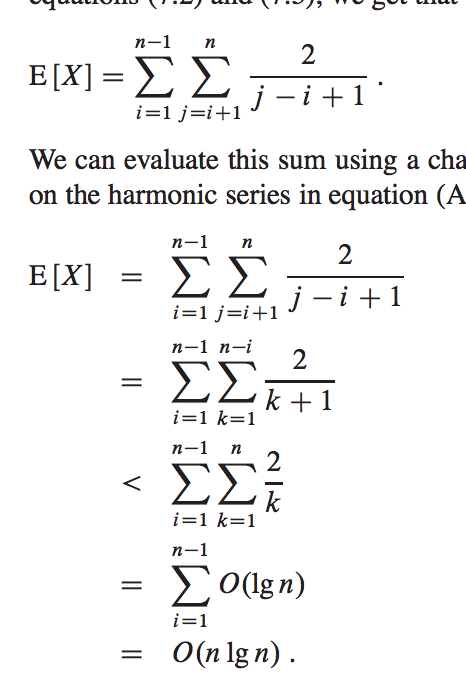
randomize-quick-sort: same time complexity





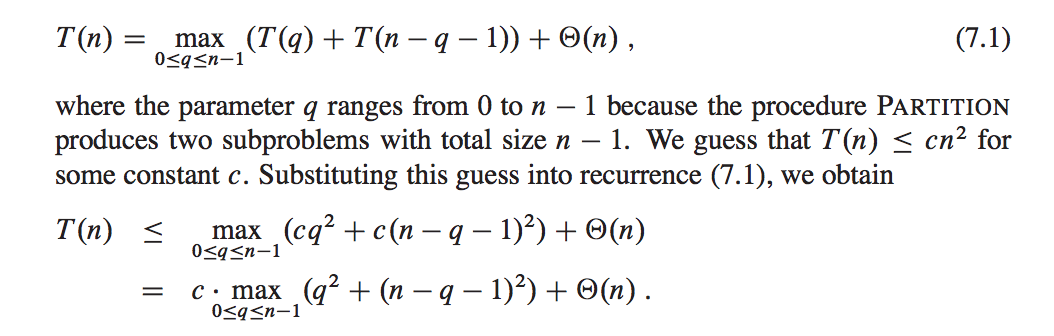
randomize 的实质: balance the recursive tree. So the depth of recursion tree is lg(n), so the total time for this is O(nlgn).

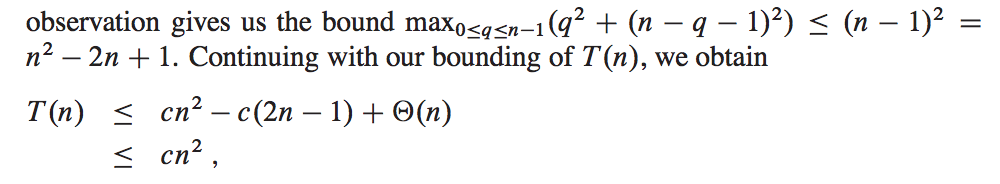




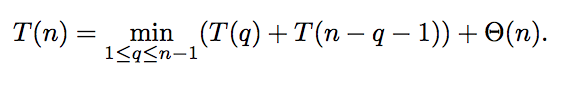
randomize-q-sort:

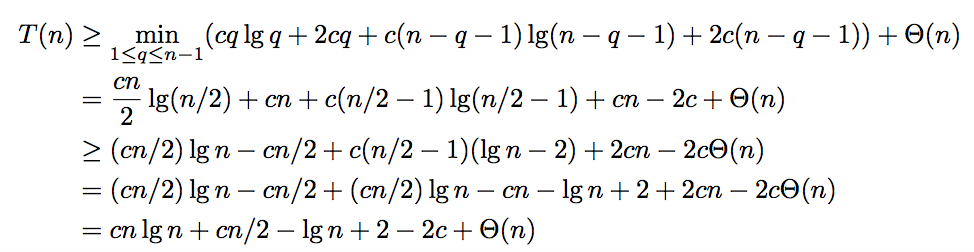
worst-case:





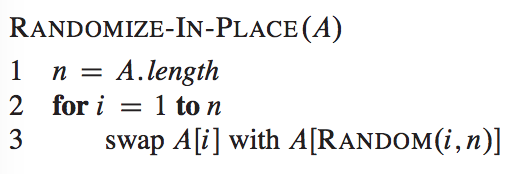
best-case, view q =n/2: Θ(nlgn):

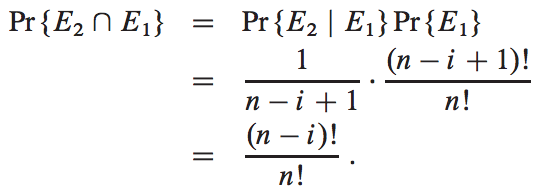




排列A(n,m)=n×（n-1）....(n-m+1) =n! /(n-m)!

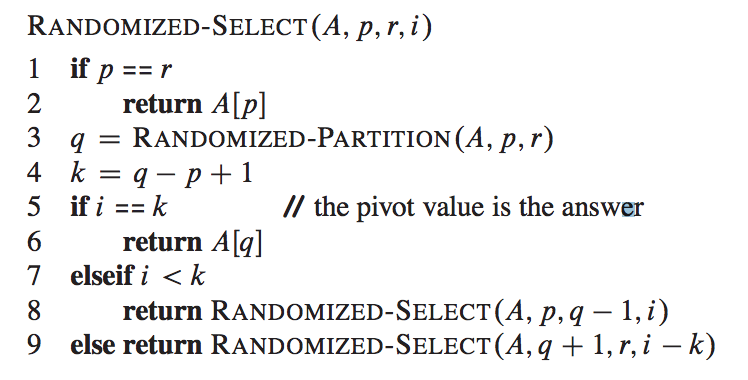
组合:C(n,m)=P(n,m)/P(m,m) =n!/m! \*(n-m)！

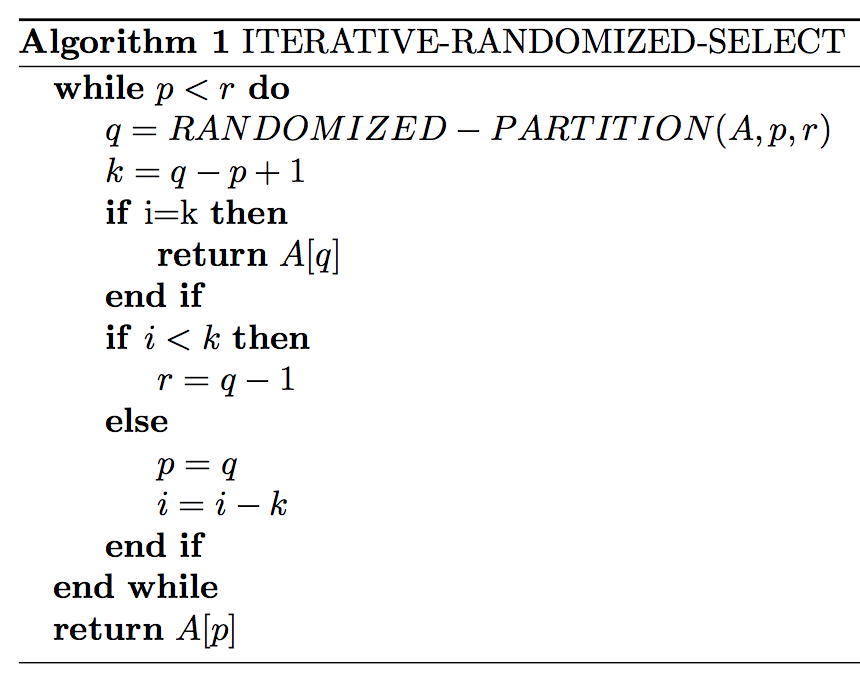


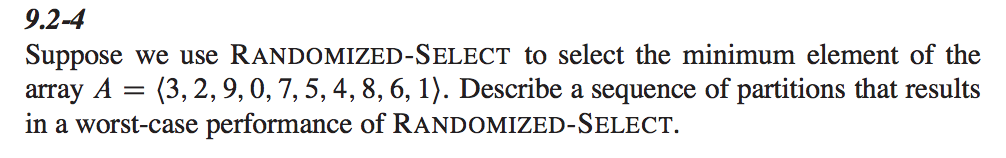
proof: e1 = the previous loop, e12= ith iteration  


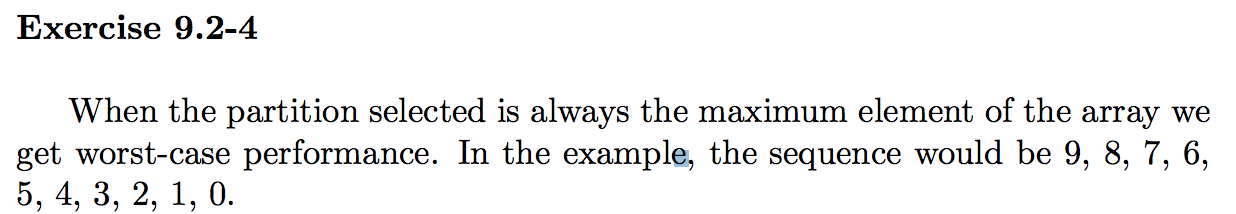
when using with-all = you got 27 possible end states, but only 6 possible ordering, so probability is not equal.

Find ith element:









hat-check problem: someone get his hat back:

