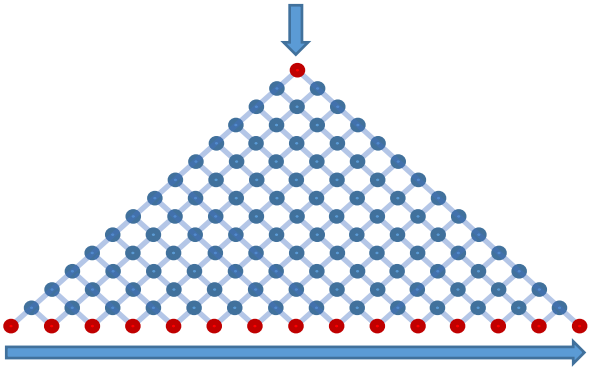
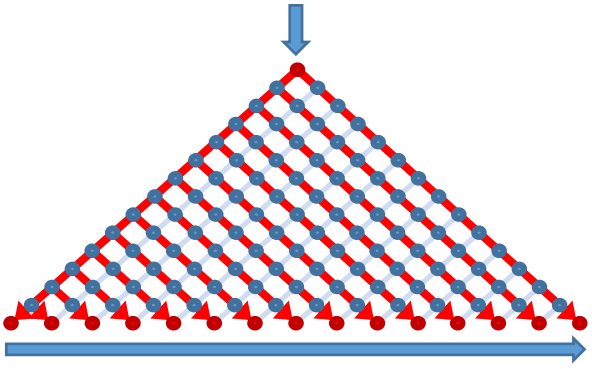
TCSS543A – Autumn 2019
Homework 2 (30 points)

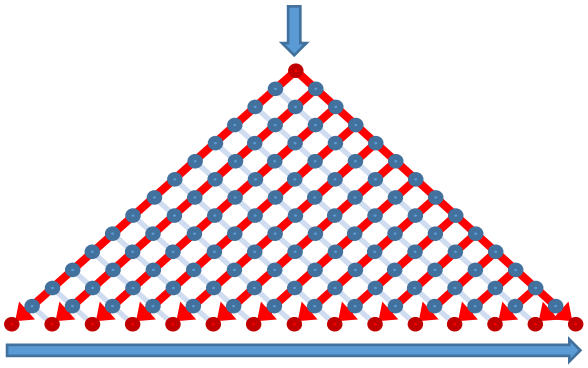
1. (5 points) Kleinberg and Tardos: Chapter 4, Exercise 3.

2. (5 points) Consider the problem of, starting the traversal at the top node, visiting all $n > 0$ bottom nodes (leaves) of a diagram similar to (but possibly much larger than) the following in left-to-right order:



Suppose that the cost of traversing a left-pointing edge is $p > 0$ and the cost of traversing a right-pointing edge is $q > 0$. There are many traversal strategies, as the examples shown in the following two diagrams:

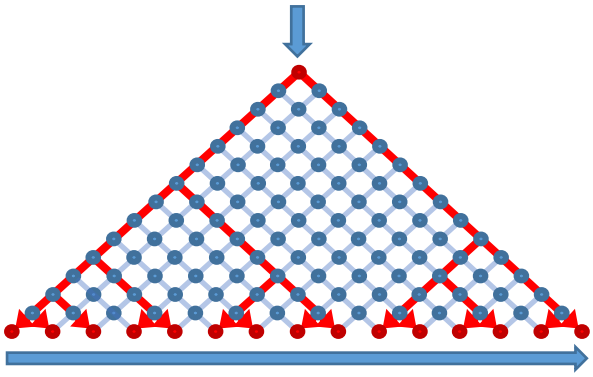




The two diagrams above represent traversals where more than half the edges are traversed (namely, all right edges and some left edges in the left diagram, and all right edges and some left edges in the right diagram. However, some strategies minimize the total number of traversed edges, and hence minimize the total cost, as the example shown in the following diagram:

Design an efficient (polynomial time) algorithm that, given n, p, and q as inputs, determines the *minimum cost* among all possible traversal strategies. Your algorithm must make no assumption about n, p, and q, except that they are all positive. *Hint: consider how a traversal splits the original tree into left and right subtrees, then express the cost in terms of the number of edges to traverse from the root to each subroot, and finally try a dynamic programming approach.*

3. (5 points) Let $G = (V, E)$ be a graph on which each edge $e \in E$ has an associated value $0 \leq rel(e) \leq 1$ that represents the reliability of a communication channel through edge e. In other words, if $e = (u, v)$, then $rel(e)$ is the probability that the channel from u to v will *not* fail.



Assume that these probabilities are independent. Give an efficient algorithm to find the most reliable path between two given vertices, and prove that it is correct.

4. (5+5+5 points) [Practical exercise] Assume A and B are two sequences (arrays) of integers.
(a) Implement (in Java) Myers' $O(ND)$ algorithm to compute the edit distance between A and B.
(b) Implement (in Java) the Wu-Manber-Myers-Miller $O(NP)$ algorithm for the same problem.
(c) Generate 8 pairs of random sequences A and B, of lengths $M = 4000$ and $N = 5000$ elements respectively, with $D = 10$, 50, 100, 200, 400, 600, 800, and 1000 randomly located deletions and $N - M + D$ insertions between them. Apply both algorithms above, then tally and report how many comparisons each algorithm performs between an element from A and an element from B.

Remarks:

- You are not required to find the longest common subsequence between A and B in either case, only their edit distance.
- To generate A and B, start by filling A with random integers, then copy A onto B, then choose D positions of B at random and delete them, then finally choose $N - M + D$ positions of B at random and insert new random values there.
- This exercise essentially reproduces the experiments reported in section 4 of the Wu-Manber-Myers-Miller work.

References:

- E. Myers, “An O(ND) difference algorithm and its variations”, Algorithmica vol. 1, pp. 251-266, Springer, 1986.
- S. Wu, U, Manber, E. Myers, W. Miller, “An O(NP) sequence comparison algorithm”, Information Processing Letters vol. 35, pp. 317-323, Springer, 1990.

5. (bonus 1+2+2 points) You are given two sequences A and B, each containing n positive integers. You can reorder each set in whatever way you want. After reordering, let a_i be the i-th element of sequence A and let b_i be the i-th element of sequence B. You then receive a payoff of $\prod_{i=1}^{n} a_i^{b_i}$ dollars. Give a greedy algorithm to maximize your payoff, prove that your algorithm is correct, and analyze its running time.