MA U34605 Quiz 05 w/e 11/12/20 ANSWERS

(1) Here is an (inefficient) procedure to sort an array of strings. Is it stable or unstable? Give reasons.

```
void insert ( int k, char * w[], char * s )
{
  int i;
  for (i=k; i>0 && strcmp(w[i-1],s) > 0; --i)
    w[i] = w[i-1];
  w[i] = s;
}

void sort ( int n, char * source[], char * target[] )
{
  int i;
  for (i=0; i<n; ++i)
    insert ( i, target, source[i] );
}
```

Answer

The 'insert' loop continues while $w[i-1]$ actually follows s lexicographically, stopping when they are equal; so s is added to the right of all strings equal to it, insert is stable, and the sort is stable.

---

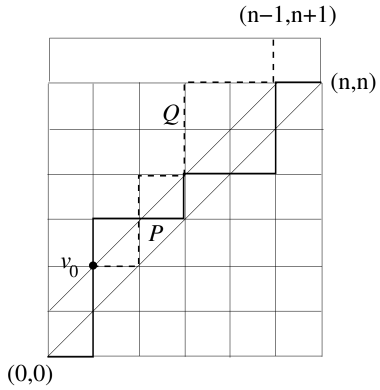
(2) Show how a 'naïve' implementation of Dijkstra's algorithm has runtime $O(m+n^2)$ with n vertices and m edges.

Answer

```
Repeated n times:
  (i) find tentative vertex u with minimum w-value (O(n))
  (ii) for all out-edges (u,v) adjust w-value of v. (O(out-degree[u]))
-------------
Overall,
(i) costs O(n^2). (ii) O(sum of out-degrees) = O(m).
```

---

(3) Dijkstra's algorithm can be improved, using some extra structure, so that after completion, for every vertex u, it is possible to produce efficiently a shortest path from s to u. Show how this can be done.



Answer

Use a `parent[]` array; initially -1 throughout. Whenever $w(v)$ is adjusted to $w(u) + w(u, v)$ when u has been made permanent, set `parent[v] = u`.

The cost remains $O(n^2 + m)$.

At the end, a shortest path from s to v can be produced in reverse by following parent links and if necessary reversed before outputting.

Cost of finding a shortest path from s to v — assuming $w[v] < \infty$ — is $O(n)$.

(4) There is an ingenious way to calculate b_n, the number of binary trees with n nodes. Let $G_{k,\ell}$ be the *grid graph* with vertices $\{(i, j) : \ 0 \le i \le k, 0 \le j \le \ell\}$ and horizontal and vertical edges of unit length. We consider paths in $G_{n,n}$. A path from $(0, 0)$ to (n, n) is *correct* if it never goes above the diagonal $i = j$, that is, for every vertex on the path, $i \ge j$. The number of correct paths is b_n: this can be shown. The trick is to show that the number of *incorrect* paths in $G_{n,n}$ equals the number of paths, unconstrained, from $(0, 0)$ to $(n-1, n+1)$ in $G_{n-1,n+1}$. The trick is: let D be the above-diagonal $\{(x, y) : \ y = x + 1\}$. Every incorrect path P must have a vertex on D. Let v_0 be the earliest vertex on the path P which belongs also to D. Let P' be the initial part of the path leading to v_0, P'' the remainder of P, Q' the result of *reflecting* P'' in D, and Q the combination of P' followed by Q'. Then Q is a path from $(0, 0)$ to $(n - 1, n + 1)$ in $G_{n-1,n+1}$. Use these facts to re-calculate b_n.

Answer

The reflection procedure maps incorrect paths in $G_{n,n}$ bijectively to arbitrary (monotone) paths from $(0, 0)$ to $(n - 1, n + 1)$ in $G_{n-1,n+1}$.

Every monotone path from $(0, 0)$ to $(n-1, n+1)$ has $2n$ edges, of which $n-1$ are horizontal, and the path is uniquely determined by identifying the horizontal edges in the path. Therefore there are

$$\binom{2n}{n-1}$$

such paths, and as many incorrect paths in $G_{n,n}$.

There are $\binom{2n}{n}$ monotone paths from $(0,0)$ to (n,n) in $G_{n,n}$. So

$$b_n = \binom{2n}{n} - \binom{2n}{n-1} = \binom{2n}{n} \times \left(1 - \frac{n}{n+1}\right) = \frac{1}{n+1}\binom{2n}{n}$$