PRINCETON UNIV SPRING '26 COS 423: THEORY OF ALGORITHMS

Homework 5

Out: *Apr 8*

**Problems:**

§1 Below you are given a list of **false statements**, construct counterexamples for them.

(a) Let $G$ be an undirected graph with *distinct* edge weights, $s$ and $t$ be two fixed vertices. There must be a *unique* $s$-$t$ shortest path.

(b) Let $G$ be a (general) undirected graph, and $M$ be a matching in $G$. Suppose there is a path $P = (v_1, v_2, \ldots, v_k)$ such that

- the length of $P$ is odd ($k$ is even);
- for all odd $i = 1, 3, \ldots, k-1$, $(v_i, v_{i+1})$ is an edge *not in* $M$;
- for all even $i = 2, 4, \ldots, k-2$, $(v_i, v_{i+1})$ is an edge *in* $M$;
- $v_1$ and $v_k$ are unmatched vertices in $M$.

Then $M$ *cannot* be a maximum matching in $G$. **Note that for bipartite $G$, this is a true statement.**

(c) The Union-Find data structure with *only* path compression, in fact, has a *worst-case* time of $O(\log n)$ per operation.

(d) The `splay` operation in the Splay Tree can be modified to the following:

```
splay(x):
   while x is not root
      rotate(x)
```

Splay Tree still has an amortized cost of $O(\log n)$.

(e) The Rabin-Karp algorithm *always* finds exactly all occurrences of the pattern in the text.

(f) Given a graph with nonnegative edge weights satisfying the Triangle inequality (i.e., $w(a, b) \leq w(a, c) + w(c, b)$), consider the following greedy algorithm for the Traveling Sales Problem:

```
start at v1
while not all vertices are visited
   find an unvisited v that is the closest from the current vertex
   move to v
move back to v1
return the tour
```

It must compute a 1.1-approximation to the optimal solution.

§2 Let $x$ be a string, if $x = \underbrace{yy\cdots y}_{r}$ for some *string* $y$, we say $x$ has repetition factor $r$.

Let $\rho(x)$ be the max $r$ such that $x$ has repetition factor $r$. Given a pattern $T$, design an algorithm that computes $\rho(T[0..i])$ for *all* $i = 0, \ldots, |T| - 1$ in linear time.

§3 Let $T$ be a pattern string in $\Sigma^n$. Design an algorithm with running time $O(n\,|\Sigma|)$ that computes for every $(i, \sigma)$ where $i \in [0, n-1]$ and $\sigma \in \Sigma$: The length of the longest prefix $\delta \leq i$ such that $T[0..\delta-1] = T[i-\delta+1..i]$ and $T[\delta] = \sigma$.

§4 A *maximal matching* is a matching that cannot increase its size by directly adding an edge, i.e., it is not a proper subset of any matching. A vertex cover is a set of vertice that contains at least one endpoint of every edge.

(a) Prove that the size of a *maximum* matching is a lower bound on the size of a minimum vertex cover.

(b) Prove that for any *maximal* matching, the set of matched vertices is a vertex cover.

(c) Show how to compute a 2-approximate minimum vertex cover in $O(m)$ time.

§5 [optional] Recall $\texttt{LCP}[i]$ is the longest common prefix of $T[\texttt{SA}[i-1]..]$ and $T[\texttt{SA}[i]..]$, where $\texttt{SA}$ is the suffix array.

(a) Suppose $\texttt{SA}[i] = k$ and $\texttt{SA}[i'] = k+1$. Prove that $\texttt{LCP}[i'] \geq \texttt{LCP}[i] - 1$.

(b) Prove that computing $\texttt{LCP}[i]$, where $\texttt{SA}[i] = k$, in the increasing order by $k$ can be done in time $O(|T|)$.