CSE521 Homework 2 Solution

Problem 1. We construct hierarchical nets N_i's such that $N_i \subseteq N_{i-1}$ for all $i \geqslant 1$ as follows. First, N_k – the highest level net – contains an arbitrary node; then each N_i for $n-1 \geqslant i \geqslant 0$ are constructed recursively in two steps: (i) select all node in N_{i+1}; (ii) while there is still a node of distance at least 2^i from all the selected node, select it. It is easy to see that this construction works.

Now, for each $x \in N_i \backslash N_{i+1}$, there must be some $y \in N_{i+1}$ such that $d(x,y) \leqslant 2^{i+1}$. We let $p(x) = y$, and say that x is a child of y and y is the parent of x. The data structure we will maintain are the hierarchical nets and this parent-child relationship. For convenience, we will maintain the link from each node to all of its children. This can at most double the space required, which is clearly linear in the size of the metric X.

Note that we do not store the lists $L_{x,i}$'s – this is why we can use only a linear amount of space. Instead, we will construct the lists $L_{q,i}$'s "on-the-fly" when we find q. First $L_{q,k+1}$ contains exactly one point – the only point in N_k. Then the following lemma gives us a relationship between $L_{q,i}$ and $L_{q,i-1}$ which we will utilize to compute $L_{q,i-1}$ from $L_{q,i}$.

Lemma 1 *For any* $y \in L_{x,i-1}$*, either* $y \in L_{x,i}$ *or* $p(y) \in L_{x,i}$.

Proof. Recall that $y \in L_{x,i-1} = B(x,2^i) \cap N_{i-2}$. Consider two cases

1. $y \in N_{i-1}$. Then $y \in B(x,2^{i+1}) \cap N_{i-1} = L_{x,i}$ because $B(x,2^i) \subseteq B(x,2^{i+1})$.

2. $y \in N_{i-2} \backslash N_{i-1}$. Then $p(y) \in N_{i-1}$ and $d(y,p(y)) \leqslant 2^{i-1}$ by the definition of $p(y)$. We have:

$$\begin{aligned} d(x,p(y)) &\leqslant d(x,y) + d(y,p(y)) && (1)\\ &\leqslant 2^i + 2^{i-1} && (2)\\ &\leqslant 2^{i+1} && (3) \end{aligned}$$

where (1) is the triangle inequality in metric X, and (2) follows from the fact that $y \in B(x,2^i)$ and the definition of $p(y)$. Thus, $p(y) \in B(x,2^{i+1})$, which means it is in $L_{x,i} = B(x,2^{i+1}) \cap N_{i-1}$.

■

Let $S_{q,i} = \{y \in N_{i-1} | p(y) \in L_{q,i+1}\} \cup L_{q,i+1}$; then the lemma implies that $L_{q,i} \subseteq S_{q,i}$. Thus, we can compute $L_{q,i}$ by first enumerating all elements of $S_{q,i}$ using the links from the points in $L_{q,i+1}$ to their children, then removing those elements that are not in $B(q,2^{i+1})$. Since $|L_{q,i+1}|$ is a constant (at most $[\lambda(X,d)]^3$), this could be done in constant time if the number of children of each node is a constant. However, this may not be the case.

To achieve constant running time, we first define the *true level* of a point y, denoted by $\ell(y)$, to be the largest i where $y \in N_i$. Now, let $S^*_{q,i} = \{y | \ell(y) = i-1 \text{ and } p(y) \in L_{q,i+1}\} \cup L_{q,i+1}$. We claim that $L_{q,i} \subseteq S^*_{q,i}$. To do so, we only need to prove that any point in $S_{q,i} \backslash S^*_{q,i}$ is not in $L_{q,i}$. Consider such a point x. Then $\ell(x) \geqslant i$, so $x \in N_i$. But since $x \notin S^*_{q,i}$, we have $x \notin L_{q,i+1}$. Thus, $x \notin B(q,2^{i+1})$, which means $x \notin B(q,2^i)$. Hence, $x \notin L_{q,i}$ as required.

With this, we can compute $L_{q,0}$ – which will give us the answer – in time $O\left(\sum_{i=0}^{k+1} |S^*_{q,i}|\right)$ by enumerating and pruning from $S^*_{q,i}$ at each step. To show that this running time is $O(k)$, we only need to show that each $S^*_{q,i}$ is of constant size. As argued above, this reduces to proving that each node has at most a constant number of children of any true level – we can easily store the children of a node so that retrieving the list of children of a certain true level can be done in time proportional to the number of children retrieved.

Consider a point x and its children $y_1, y_2, \dots y_t$ such that $\ell(y_j) = i$ for all j. Then all $y_1, y_2, \dots y_t$ are in the ball $B(x, 2^{i+1})$. $B(x, 2^{i+1})$ can be covered by $\lambda(X, d)^2$ balls of radius 2^{i-1}, none among which contain two points among $y_1, y_2, \dots y_t$, for otherwise the distance between these two points would be smaller than 2^i. Thus, $t \leqslant \lambda(X, d)^2$ as required. This completes the proof.

Problem 2.

1. Let $g(n) = 2^{\lfloor \log n \rfloor}$, i.e. $g(n)$ is the largest power of 2 that is not bigger than n. Then we define the potential function $\Phi_i = 2(i - g(i))$. Clearly $\Phi_1 = 0$ and $\Phi_i \geqslant 0$. We claim that $\Phi_i - \Phi_{i-1} + f(i) \leqslant 3$ for all i. This will shows that the amortized cost of each operation is $O(1)$.

   Consider two cases

   (a) $i = 2^k$ for some k. Then $\Phi_i = 0$, $\Phi_{i-1} = 2(2^k - 1 - 2^{k-1})$ and $f(i) = 2^k$. Hence,
   $$\Phi_i - \Phi_{i-1} + f(i) = -2(2^{k-1} - 1) + 2^k = 2. \tag{4}$$

   (b) $2^k < i < 2^{k+1}$ for some i. Then $i - 1 \geqslant 2^k$. Thus, $\Phi_i = 2(i - 2^k)$, $\Phi_{i-1} = 2(i - 2^k - 1)$ and $f(i) = 1$. Hence,
   $$\Phi_i - \Phi_{i-1} + f(i) = 2 + 1 = 3. \tag{5}$$

2. We assume that allocating new hash table takes $O(1)$ time and let k_i and n_i be the number of elements in the hash table and the hash table's size at time i respectively. Then define
   $$\Phi_i = \begin{cases} 0 & \text{if } \frac{3n_i}{8} \leqslant k_i \leqslant \frac{n_i}{2} \\ 3\left(k_i - \frac{n_i}{2}\right) & \text{if } k_i > \frac{n_i}{2} \\ 2\left(\frac{3n_i}{8} - k_i\right) & \text{if } k < \frac{3n_i}{8} \end{cases} \tag{6}$$

   With this, checking that for any $i \geqslant 1$, $\Phi_i - \Phi_{i-1} + c(i) \leqslant C$ for some constant C where $c(i)$ is the cost of the ith operation is a matter of case by case analysis and algebra.

   However, there's still one delicate point: the total cost of a sequence of i operations is upperbounded by $Ci - \Phi_i + \Phi_0$, which is not immediately linear in i since $\Phi_0 \neq 0$. Therefore, in order to conlude that the amortized cost of each operation is $O(1)$, we need to establish that $\Phi_0 - \Phi_i \leqslant iD$ for some constant D. We show this by considering two cases.

   - $i \leqslant \frac{3n_0}{8}$; then $n_i = n_0$ and $k_i = i$. Thus $\Phi_0 - \Phi_i = k_i = i$.
   - $i \geqslant \frac{3n_0}{8}$; then obviously $\Phi_0 \leqslant i$.

   This completes the proof.

3. Assume that the cost of each stack operation is 1 and let S_i and S_o denote the size of the in and out stacks respectively. Then, define $\phi_i = 2S_i$. Clearly $\phi_0 = 0$ and $\phi_i \geqslant 0$ for all i. There are three kind of operations: the insertions, the "normal" deletions which do not cause a stack transfer and the "bad" deletions that cause a stack transfer. One can easily check that $\phi_i - \phi_{i-1} + c_i \leqslant 2$. Therefore, the amortized cost of each operation is $O(1)$.

Problem 3.

1. Let $h_f(i) = d_f(s, i)$ be the height of i in the BFS tree of G_f and $(s = p'_1)p'_2p'_3 \cdots (p'_k = v)$ be the shortest path from s to v in $G_{f'}$.

   First, note that for any edge $ij \in G_f$, $|h_f(i) - h_f(j)| \leqslant 1$. In addition, for any edge $i'j' \in G_{f'}$, either $ij \in G_f$ or $ji \in G_f$. Thus for any edge $i'j' \in G_{f'}$, $|h_f(i') - h_f(j')| \leqslant 1$. This implies $\sum_{i=1}^{k-1} |h_f(p'_i) -$

$h_f(p'_{i+1})| \leqslant k$. On the other hand $\sum_{i=1}^{k-1} |h_f(p'_i) - h_f(p'_{i+1})| \geqslant |h_f(p'_1) - h_f(p'_k)| = h_f(v)$. Therefore, $k \geqslant h_f(v)$, which means the shortest path from s to v in $G_{f'}$ is at least as long as the shortest path from s to v in G_f.

2. Assume that uv is the bottleneck link on the shortest augmenting path of f and f''; then after augmenting f, the edge uv is replaced by the edge vu. Thus, there must be another step where we vu is replaced by uv. We denote the flow being augmented at this step f'. Now, note that when augmenting a flow f^*, for any edge ij on the shortest augmenting path, we have $h_{f^*}(j) = h_{f^*}(i) + 1$. Thus, we have

$$h_f(v) = h_f(u) + 1 \tag{7}$$
$$h_{f'}(u) = h_{f'}(v) + 1 \tag{8}$$

These equations, together with the facts that $h_{f''}(i) \geqslant h_{f'}(i) \geqslant h_f(i)$ for all i, imply that $h_{f''}(u) \geqslant h_f(u) + 2$.

3. Each time an edge uv is the bottleneck link of an augmentation, $d_f(s,u)$ increase by 2. In addition, $d_f(s,u)$ never decreases. Furthermore, $d_f(s,u) \leqslant |V|$ for all u and f. Thus, each edge can be the bottleneck link $O(|V|)$ time. There are $|E|$ edges, and each augmentation has at least one bottleneck links. Therefore, the number of augmentation is $O(|V| \cdot |E|)$.