University of Waterloo
CS341 - Winter 2016
Assignment 3

Due Date: Thursday Mar 3 at 11:59pm

Problem 1 Snake and Ladder

Ans:
code in a3q1.cpp

Problem 2 Huffman coding

Ans:
(a)
Assume that the frequencies do not add to 1.
As we know, "If some character occurs with frequency more than 2/5, then there is guaranteed to be a codeword of length 1." can be implied to "If a codeword is guaranteed to be 1 bit, then there is no frequency can be greater than 2/5".

Without loss of generality (WLOG), the Huffman tree should be like:
.   a+b+c+d
.   /   \
.   a+b   c+d
.   /   \   /   \
. a   b   c   d
(a+b+c+d=1)
Now, we should prove that a, b, c and d are no larger than 2/5.

WLOG $a = b \le c \le d$, we can get:
.   2a+c+d
.   /   \
.   a+a   c+d
.   /   \   /   \
. a   a   c   d
Then, we can find the maximal d that is consistent with the Huffman tree. we can get:

- $a \le c$
- $a \le d$
- $c \le 2a$
- $d \le 2a$

Then make $d = 1 - c - 2a$:

- $a \leq c$
- $a \leq \frac{1-c}{3}$
- $c/2 \leq a$
- $\frac{1-c}{4} \leq a$

Finally, we can get $\frac{c}{2} \leq a \leq \frac{1-c}{3}$,
so $\frac{c}{2} \leq \frac{1-c}{3}$, implies to, $c \leq \frac{2}{5}$.
and $a = b \leq c \leq \frac{2}{5}$, $d = 1 - c - 2a \leq 1 - \frac{2}{5} - \frac{4}{5} \leq \frac{2}{5}$.
So, If a codeword is guaranteed to be 1 bit, then there is no frequency can be greater than 2/5.
We can get: If some character occurs with frequency more than 2/5, then there is guaranteed to be a codeword of length 1.

(b)
As we know, at least 4 symbols in the alphabet. Assume by contradiction that there is a codeword of length 1. The codeword of length 1 will branch from the root of the tree. WLOG, it has frequency f. It has to be merged with a subtree, that results 2 subtrees (it may be leaves) by the construction of Huffman tree. Because this last element was pulled from the priority queue, both 2 subtrees have to have cumulative weight less than f. Then, because $f < \frac{1}{3}$ by assumption all characters occur with frequency less than $\frac{1}{3}$,
(weight of 2 subtrees) + f < 1.
Since the sum of all weights in the subtree has to be 1, this is a contradiction. So, there is guaranteed to be no codeword of length 1.

Problem 3 Thickest Paths

Ans:
Use Dijkstra's Algorithm to solve the problem:
pseudocode:
1. $dist(t) = \infty$ for every $t \in V$.
2. $dist(s) = 0$
3. max-count[V-1]=0 for all t vertices, count the maximum of minimum thickness
4. Q=make-priority-queue(V), all vertices are put in the priority queue, using dist[t] as the key (priority) value of t.
5. while Q is not empty do
6.   u=delete-min(Q), dequeue the vertex with minimum dist-value
7.   for each out-neighbour t of u
8.   if $dist[u] + l_{ut} < dist[t]$
9. $dist[t] = dist[u] + l_{ut}$

```
        if dist[t] > max-count[t]
            max-count[t]=dist[t]
        decrease-key(Q,t) and parent[t] = u
output all paths with max-count.
```

Proof of Correctness:
Let $\delta(t)$ denote the true thickest path distance of vertex t from the source s.Observe that Dijkstras algorithm works by estimating an initial thickest path distance of ∞ from the source and gradually lowering this estimate.

- If $dist[t] = \delta(t)$ for any vertex t, at any stage of Dijkstras algorithm, then $d[t] = \delta(t)$ for the rest of the algorithm.
  Proof: clearly, $d[t]$ cannot be smaller than $\delta(t)$; likewise, the path of s,t cannot be constructed.

- Let $\{t_1 = s,\ t_2,...,t_{|V|}\}$ denote the sequence of vertices dequeued from the heap Q, by Dijkstras algorithm, when vertex t_i is dequeued from Q, $d[t_i] = \delta(t_i)$.
  Proof: WLOG, we assume that each vertex is reachable from the s through a finite length path or an arc of length infinity.
  Obviously, the claim is true for $t_1 = s$, since $d[s] = \delta(s) = 0$ and all edge weights are positive.
  Assume that the claim is true for the first $k-1$ vertices, i.e., assume that for each $i = 2, 3, ..., k-1$, when vertex t_i is dequeued from Q, $d[t_i] = \delta(t_i)$.
  Use induction, we can get that if the thickest path from $t_1 = s$ to t_k consisted of vertices from the set $S = \{t_1, ..., t_{k-1}\}$, then $d[t_k] = \delta(t_k)$

Time complexity: All the enqueue, dequeue and update operations can be implemented in $O(\log|V|)$ time using heap, and thus the total time is $O((|V| + \sum_t outdeg(t))\log|V|) = O((|V| + |E|)\log|V|)$.

Problem 4 Maximum Interval Coloring

Ans:
Use Greedy's Algorithm to solve the problem:
pseudocode:

```
Let earliest starting time is min_i s_i
Let earliest finishing time is min_i f_i
sort the intervals by the finishing time so that f_1 < f_2 < ... < f_n
color-count = 0
for 1 ≤ i ≤ n do
    if interval i's s_i, s_i ≤ f_{i-1} (there is a conflict), then
        color-count += 1
        if color-count > k, then
            mark this interval color "X"
```

10. else if color-count <= k
11. mark this interval a new color inside k colors
12. else if interval i's s_i, $s_i > f_{i-1}$ (there is no conflict)
13. find the color of interval whose finishing time is most close to the s_i, and mark this color for interval i
14. color-count = 0

Proof of Correctness:
Claim: There are colors less than or equal to k.
Proof: For line 8 in the algorithm, we can get any intervals which wants to use k+1 color will be marked as "X", so there is no k+1 color. So there is no extra colors exceeding k.
Claim: There exists an optimal solution with interval i.
And $\{s_{i_1}, f_{i_1}\}, \{s_{i_2}, f_{i_2}\}, \{s_{i_k}, f_{i_k}\}, \{s_{j_{k+1}}, f_{j_{k+1}}\}, ..., \{s_{j_l}, f_{j_l}\}$ is an optimal solution with $f_{i_k} \leq f_{j_k}$
Proof: The base case holds. Assume the claim is true for $c \geq 1$, and then we can prove it out by induction.

Time Complexity: Because the for loop in my algorithm runs for n times. And for the inner part, the comparison recurs between current intervals and the previous intervals. Clearly, n intervals can be looked as n nodes in a heap, so the inner operation time should be $\log n$. Therefore, the total time is $O(n \log n)$.