Summary on Lecture 12, August 5, 2015

Prefix Codes

We choose the prefix code 00, 01, 100, 1010, 1011, 11. This set could serve as a code for the letters in an alphabet $\Sigma = \{a_1, a_2, a_3, a_4, a_5, a_6\}$ that has six letters. It is the set of leaves of a labeled binary tree in Fig. 1.

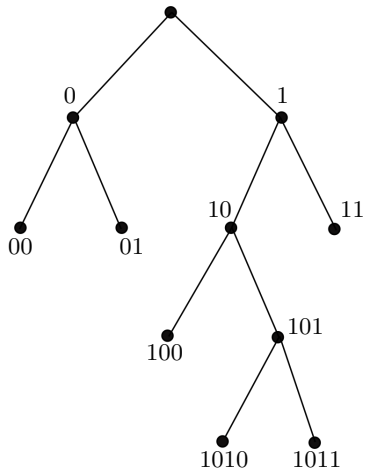

Fig. 1. Prefix code. Here $a_1 := 00$, $a_2 := 01$, $a_3 := 100$, $a_4 := 1010$, $a_5 := 1011$, $a_6 := 11$.

Now assume that we are given the binary string

11101011011000100111110010

We visit vertex 1, then vertex 11. Since vertex 11 is a leaf, we record 11 and return to the root. Next we visit vertices 1, 10, 101, and 1010. Since 1010 is a leaf, we record 1010 and return again to the root. Proceeding in this way, we obtain the sequence of code symbols

11, 1010, 11, 01, 100, 01, 00, 11, 11, 100, STOP

and have 10 left over which does not give any symbol encoded by the leaves. Now we suppose that we know how frequently each letter in $\Sigma = \{a_1, a_2, a_3, a_4, a_5, a_6\}$ is used in sending messages. Namely, we assume that the frequencies 10, 10, 15, 20, 20, 25 correspond to the letters $a_1, a_2, a_3, a_4, a_5, a_6$.

Then we would like to find the most efficient prefix code, i.e. we want to minimize the average length of a code message using (say, 100 letters) from Σ. Thus all we need is an **optimal binary tree for the weights** $\{10, 10, 15, 20, 20, 25\}$. Now we run Huffman algorithm and obtain the optimal tree, see Fig. 2 (b):

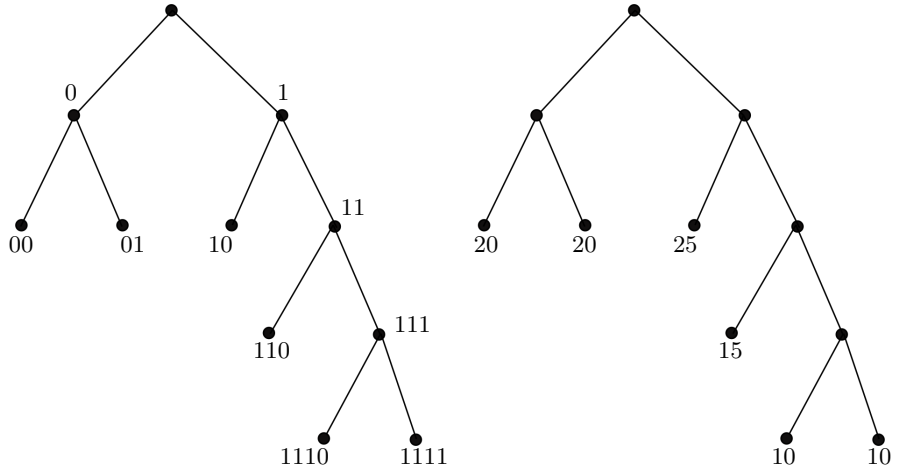

Fig. 2 (a) (b)

Then the same tree (see Fig. 2 (b)) with binary labels gives new prefix code

$$a_1 := 1110,\ a_2 := 1111,\ a_3 := 110,\ a_4 := 00,\ a_5 := 01,\ a_6 := 10.$$

This is an **optimal prefix code for given frequences.**

Example. Here is a prefix code: $\{00, 010, 0110, 0111, 10, 11\}$ We construct a binary tree corresponding to this prefix code:

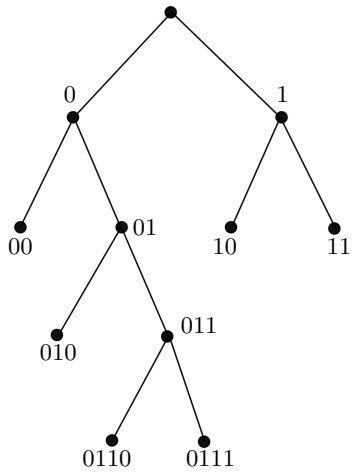

Fig. 3. Prefix code: here 00 :=A, 10 :=D, 11 :=E, 010 :=H, 0110 :=M, and 0111 :=' (the apostrophe)

Now we decode the string

$$0101101100010110110110110010 0$$

to get the string HEMADEMEMAD (he made me mad).

Exercise 1. Decode the string:

$$1000100100010011000100000110.010110110001011011000011000 10.011000011000101110001010110010.$$

Exercise 2. Assume that the letters A,D,E,H,M,' have frequences 30, 25, 27, 20, 33, 10 respectively. Find an optimal prefix code and encode the message "HEMADEMEMAD".

Digraphs and shortest paths. Let $G = (V, E)$ be a graph or digraph. We say that G is *weighted* if we are given a function $\mathsf{wt} : E \to (0, \infty)$. In other words, each edge $e \in E$ is given a positive weight $\mathsf{wt}(e)$. It is convenient to have a convention that if $v, v' \in V(G)$ are not connected by an edge, then a *virtual edge* (v, v') has weight ∞. We also accept a convention that a *virtual edge* (v, v) has zero weight. Then once we have a weighted graph, it makes sense to determine a *shortest distance* from one vertex to another.

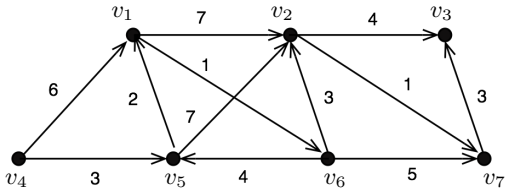

Fig. 4. Weighted digraph G_1.

Consider the digraph G_1 in Fig. 4. It is easy to check that the shortest distance (weight) from v_4 to v_1 is 5 when we take the route $v_4 \to v_5 \to v_1$, and it is shorter than a "direct" shot $v_4 \to v_1$.

Now the "shortest" paths $v_4 \to v_1 \to v_2 \to v_3$ and $v_4 \to v_5 \to v_2 \to v_3$ from v_4 to v_3 have weights $6+7+4 = 17$ and $3 + 7 + 4 = 14$, respectively, but the "longer" path $v_4 \to v_5 \to v_1 \to v_6 \to v_2 \to v_7 \to v_4$ has weight $3 + 2 + 1 + 3 + 1 + 3 = 13$, which is less than either of these. Thus length is not directly related to weight.

There are special vertices in G_1, namely, v_4 is a *source*, and v_3 is a *sink*. Indeed, v_4 has only *outbounded* edges, and v_3 only *inbounded* edges. In the case when a weighted digraph G has a source and a sink (and no loops and parallel edges), it is called a *scheduling network*. The objective here is to find a shortest path (i.e., with minimal weight) from a source to a sink. This is known as *scheduling problem.*