Algorithms and Data Structures II

Lecture 8:

Algorithm Design Techniques: Greedy Algorithms

https://elms.u-aizu.ac.jp

greedy algorithm

- Algorithms for optimization problems typically go through a sequence of steps, with a set of choices at each step.
- A greedy algorithm always makes the choice that looks best at the moment. That is, it makes a locally optimal choice in the hope that this choice will lead to a globally optimal solution.

Examples of greedy algorithms

- Examples of greedy algorithms
  - Prim's algorithm and Kruskal's algorithm for the minimum spanning tree problem
  - Dijkstra's algorithm for the single source shortest path problem.
- Now, we introduce another application of greedy algorithms, the design of data-compression (Huffman) codes.

File Compression

- Assume that the 26 characters A,B,...,Z are assigned the 5-bits binary codes as follows: (A,00001)(1), (B,00010)(2), (C,00011)(3), ..., (Z,11010)(26).
- Given ABRACADABRA, using the 5-bits binary representation, it is encoded as the following:

ABRACADABRA

000010001010010000010001100001
0010000001000101001000001

- To decode this message, read off five bits at a time and convert according to the above assignment.
- In the sequence, D appears once and A appears 5 times, while they are both assigned 5 bits in the binary codes.

shorten the sequence

- An idea to shorten the sequence is to assign the shortest bit-strings for the most frequently appeared characters.
- In “ABRACADABRA” A appears 5 times, B appears twice, C appears once, D appears once, and R appears twice.

shorten the sequence(contd.)

- We may give an assignment like (A,0), (B,1), (R,01), (C,10), (D,11), and ABRACADABRA is encoded as 010101001101010
- This uses 15 bits compared with the 55 bits before. But, this code depends on the blanks to delimit the characters. Without the blanks, the string 010101001101010 could be decoded as RRRARBRRA or as several other strings.

shorten the sequence(contd.2)

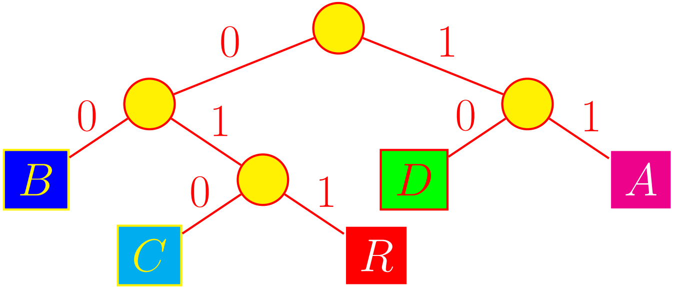
- How to encode the string compactly without using the delimiters.
- If the code of a character is not the prefix of any code of other characters then delimiters are not needed.
  - 0, 00, 001, and 0011 are prefixes of 00110.
  - In (A,0), (B,1), (R,01), (C,10), (D,11), the code of A is a prefix of R, the code of B is a prefix of C and D.

shorten the sequence(contd.3)

- A binary tree with M leaves can be used to encode any message with M different characters (one leaf holds one character).
- The code for each character is determined by the path from the root to the leaf that holds that character, with 0 for go 'left' and 1 for go 'right'.

shorten the sequence(contd.4)

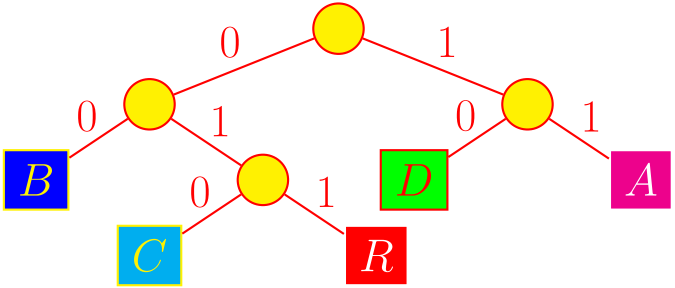
- For example, the following binary tree encodes A as 11, B as 00, C as 010, D as 10, and R as 011.



$ABRACADABRA$ is encoded as
1100011110101110110001111 25 bits

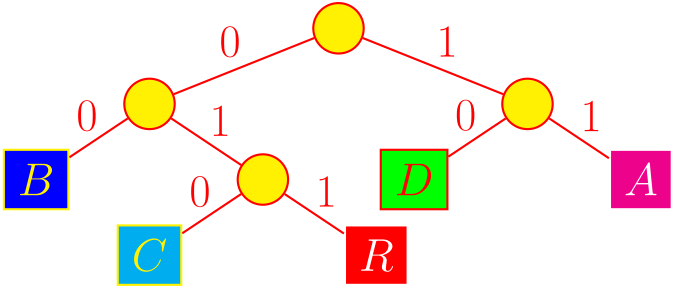
shorten the sequence(contd.5)

- The binary tree representation guarantees that no character code is the prefix of another, so the string is uniquely decodable from the tree.



shorten the sequence(contd.6)

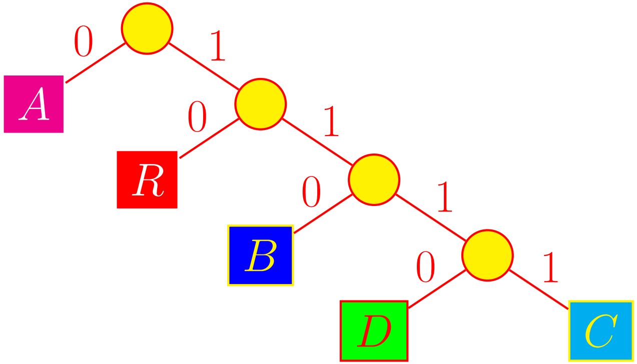
- Starting at the root, proceed down the tree according to the bits of the message: each time a leaf is encountered, output the character at that leaf and restart at the root.



1100011111010111101110001111

shorten the sequence(contd.7)

- The following is another binary tree for encoding $ABRACADABRA$.



01101001111011100110100 23 bits

Huffman encoding

- Question: which binary tree is the best one to use (gives the shortest binary sequence for a given string of characters).
- D. Huffman gave an elegant method (called Huffman encoding) to compute a binary tree which leads to a bit string of minimal length for any given string of characters.

Huffman Encoding

- Huffman encoding uses a greedy approach to find a binary tree for encoding.
- Let S be the set of characters that appear in a given string.

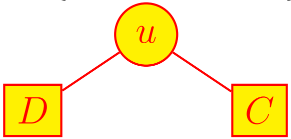
Huffman Encoding steps

- Step 1, count the frequency of each character of S in the given string. Consider S as a set of roots of binary trees, each tree has one node.
- Step 2, find two roots with the minimum frequencies and generate a new node as the root of the two roots. The new root has the frequency of the sum of frequencies of its two sons. Delete the two sons from S and add the new root to S. Repeat Step 2, until $|S| = 1$.

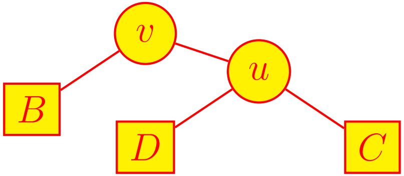
Huffman Encoding steps

- Given string $ABRACADABRA$. $S = \{A, B, C, D, R\}$, $w(A) = 5, w(B) = 2, w(C) = 1, w(D) = 1, w(R) = 2$.

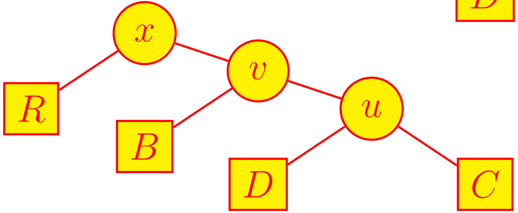
$$S = \{\overset{5}{A}, \overset{2}{B}, \overset{1}{C}, \overset{1}{D}, \overset{2}{R}\}$$



$$S = \{\overset{5}{A}, \overset{2}{B}, \overset{2}{u}, \overset{2}{R}\}$$

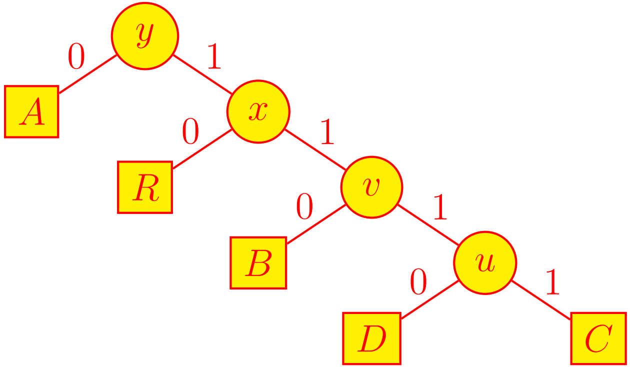


$$S = \{\overset{5}{A}, \overset{4}{v}, \overset{2}{R}\}$$



Huffman Encoding steps

$S = \{\overset{5}{A}, \overset{6}{x}\}$



$S = \{\overset{11}{y}\}$

- We assign the binary codes to the characters of S as: $(A, 0)$, $(B, 110)$, $(C, 1111)$, $(D, 1110)$, and $(R, 10)$.
  01101001111011100110100 23 bits

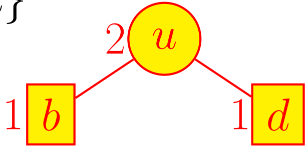
Another Example

- Given string "$aaabccdef$". $S = \{a, b, c, d, e, f\}$, $w(a) = 3$, $w(b) = 1$, $w(c) = 2$, $w(d) = 1$, $w(e) = 1$, and $w(f) = 1$. Using the greedy approach, we get a binary tree as shown on the next slide.

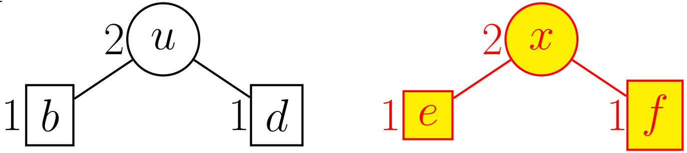
Another Example(contd)

$$S = \{a, b, c, d, e, f\}$$

$$S = \{a, c, e, f, u\}$$

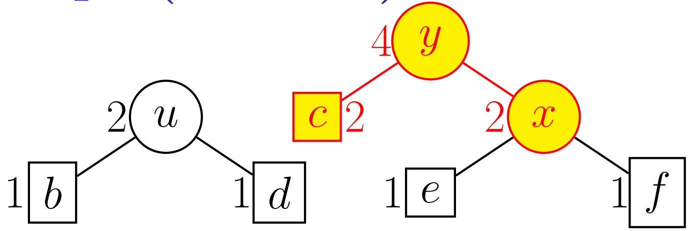


$$S = \{a, c, u, x\}$$

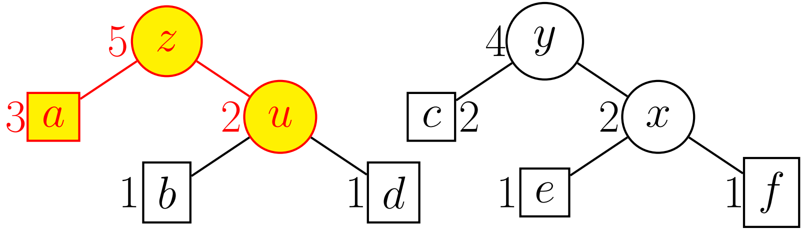


Another Example(contd 2)

$S = \{a, u, y\}$

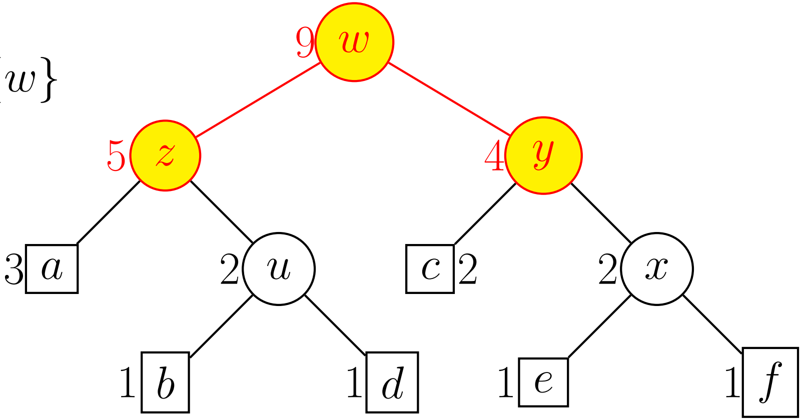


$S = \{y, z\}$

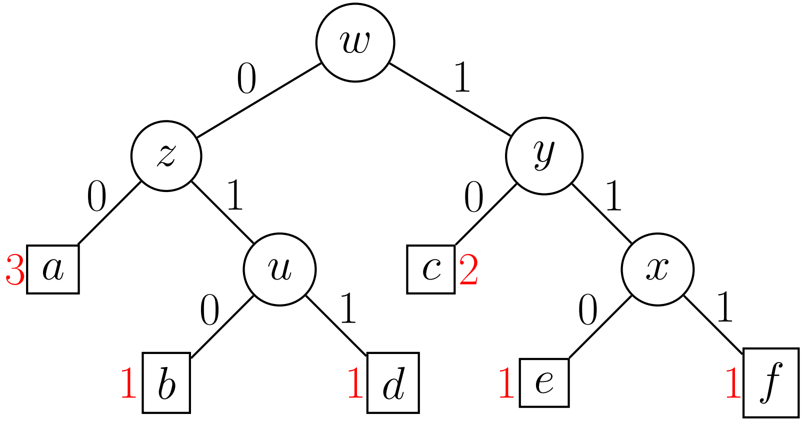


Another Example(contd 3)

$$S = \{w\}$$



Another Example(contd 4)



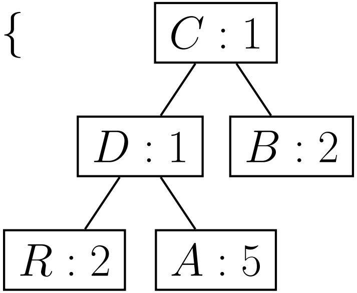
We assign the binary codes to the characters of S as:
(a,00), (b,010), (c,10), (d,011), (e,110), (f,111).

Another Example(contd 5)

- In finding the binary tree for encoding, we repeatedly find two roots with the minimum frequencies from the set S and delete the found roots from S, and then make a new root to insert to S.
- Obviously, a data structure of heap is convenient for those operations.

Recall: downheap

```
void downheap(int k, int n){
  int j,u,v;
  u=heap[k];v=index[k];
  while (k<=n/2){
    j=k+k;
    if (j<n && heap[j]>heap[j+1]) j++;
    if (u<=heap[j]) break;
    heap[k]=heap[j]; index[k]=index[j];
    k=j; }
  heap[k]=u; index[k]=v;}
```


$C:1$
$D:1$
$B:2$
$R:2$
$A:5$