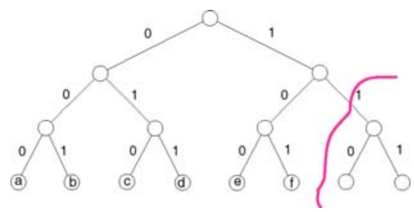
Lecture 13, Nov 6 2014

Huffman encoding

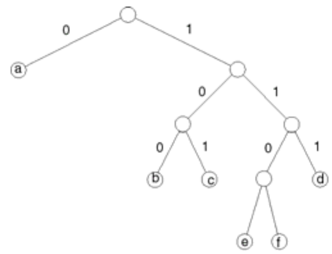
- Idea: represent often encountered letters by shorter codes.
- Prefix code: a code for x is not a prefix for any code-word for y.



- In this example: c=010, e=100

Huffman encoding

- Assume that a is a very common symbol.

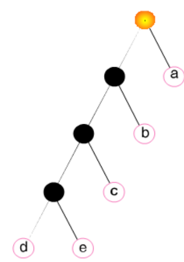


- Now: a = 0
  b = 100
  e = 1100

Huffman encoding

- Assume we know symbol frequencies (%):

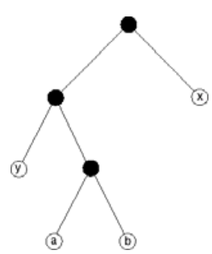
50	40	5	3	2
a	b	c	d	e



- 50*1+40*2+5*3+3*4+2*4 = 165, 1.65b/symbol instead of 3 !
- Close to Shannon's limit $-\sum_i p_i \log_2 p_i \approx 1.51,\quad p_i$=prob of letter i

Generating optimum encoding

- Claim: Let x&y be lowest freq. characters.
  Then there exists an optimum code where x&y differ only in 1 last bit.
- Consider a and b being the "deepest" symbols in the tree sharing parent.
  (Is it possible that deepest symbol does not have a sibling ?)

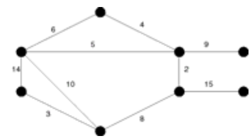


$$\left.\begin{array}{ll} WLOG & f(a)\le f(b),\ f(x)\le f(y) \\ also: & f(x)\le f(a),\ f(y)\le f(b) \end{array}\right\}$$

$\Rightarrow$ *exchanging* $x \leftrightarrow a, y \leftrightarrow b$ can only help!

- So does this mean that we do not need any other codes ?
  - » [Hint: consider a sequence abcabcabc….]

Graphs



- A set of nodes (vertices), denoted by V.
- A set of edges, denoted by E.
- What is an *edge* ?
  - » Conceptually, an edge specifies 2 nodes (u,v) that it connects. (e.g. “Joe likes apples”)
  - » Can have associated direction. (“directed graph”) (e.g. “Joe follows Mike on Twitter”)
  - » Can have a “weight”, i.e. a real number associated with it. (e.g. “distance from address1 to address2 is 10 miles”)

Graphs - representations

- Adjacency matrix:
  A(i,j) =1 iff edge (i,j) exists, i.e. (i,j) ∈ E, otherwise A(i,j)=0
- Adjacency-list:
  For every node v∈ V, create linked list of adjacent edges.
- Easy to convert from one representation to another, but takes time.
- Sparse vs. dense graphs
- Representing weighted graphs

Types/properties of graphs

- Directed or undirected
- Weighted/unweighted
- Sparse/Dense
- Connected/ not connected
- Tree/Forest (no cycles)
- Bipartite
- Strongly connected (for directed graph)
- ...

156

Examples of graph problems

- Is there a cycle ? Directed cycle ? Negative weight cycle ?
- Can you reach from node u to node v ?
- What is the shortest path from u to v ?
- All-pairs shortest paths
- Find spanning tree (tree, subset of original edges, touches all nodes)
- Find spanning tree of minimum weight (for weighted graphs)
- Maximum Bipartite matching
- ...

- We will discuss some of the above problems later.
  Some other problems are in subsequent courses (e.g. CS261)

157

Graph Algorithms

- Examples of graph problems:
  - Direct applications:
    - City streets map: reachability, shortest path, congestion management
    - Communication networks: planning, fault tolerance/reliability, topology augmentation
  - Indirect applications:
    - Assigning interns to hospitals
    - Scheduling jobs on a multiprocessor
    - Searching solution spaces
- Restate as a graph problem ⟶ solve ⟶ map back

158

Depth First Search

- Visit(u)

```
color(u) = gray;   d(u) = time;  time++;
for each neighbor w of u:
     if w is white then Visit(w)
color(u) = black;  f(u) = time;  time++;
```

- Initially, set all nodes to be white,
  examine nodes one-by-one, call Visit if node is still white.
- Node visited once, edge touched twice:
  Running time $O(n+m)$

159

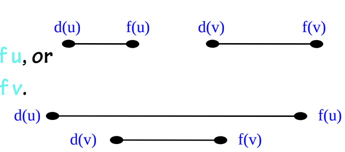
Edge Classification

- Classification of uw according to (color of u) -> (color of w):
  (when the edge is considered)
  - » Tree edge: gray -> white
  - » Back edge: gray -> gray
  - » Forward: gray -> black, u ancestor of w.
  - » Cross: other gray -> black edges.
- How to distinguish forward and cross edges ??
  - » We can use d() time !
  - » d(u) < d(w) - forward edge
- At home: create examples for each edge type.

160

Parenthesis Theorem

- Theorem:
  For any two nodes u and v,
  the two intervals [d(u),f(u)] and [d(v),f(v)] either:
  - » Do not intersect, or
  - » [d(u),f(u)] includes [d(v),f(v)] , v descendant of u, or
  - » [d(v),f(v)] includes [d(u),f(u)] , u descendant of v.



- Proof:
  - » Assume (wlog) d(u)< d(v).
  - » If v was not discovered before finishing u, then we have case 1 above.
  - » If v was discovered, then we have to finish it before returning and finishing u, leading to case 2.
  - » Case 3 is symmetric.

161