Greedy Algortihm

- Maximize local gain without considering whether it would lead to global maximum.
- A bird in hand worth two in bush.
- How can it be ensured that greedy strategy works for a problem?
- Two ingradients are exhibited by such problems
  - greedy choice property,
  - optimal substructure property.

Greedy Choice

- Greedy algorithm progresses in top-down manner, making one choice at each step.
- Iteratively reduces the given problem into smaller and smaller instances after a choice is made each time.
- A greedy choice may depend on choices already made but not on future choices to be made.
- Cleverness in applying greedy strategy lies in ensuring that local optimal leads to global optimal.

Optimal Substructure

- When an optimal solution is a superset of the optimal solution for a subproblem – then problem exhibits optimal substructure property.
- Thus, correctness of greedy algorithms can be proved in following manner
  - Examine the globally optimal solution
  - Prove that solution can be modified to pick the greedy choice at the first step
  - The above choice reduces the problem to similar but smaller instance.
  - Then apply induction to prove that greedy choice can be made at each step.

Hoffman Code

- Develops a code for each character such that most frequently used character gets smallest code.
- Used for compressing data.
- Consider a data file with 10^5 characters.

	a	b	c	d	e	f
Freq. (in 1000s)	45	13	12	16	9	5
Fixed length code	000	001	010	011	100	101
Variable length code	0	101	100	111	1101	1100

Greedy Algortihm

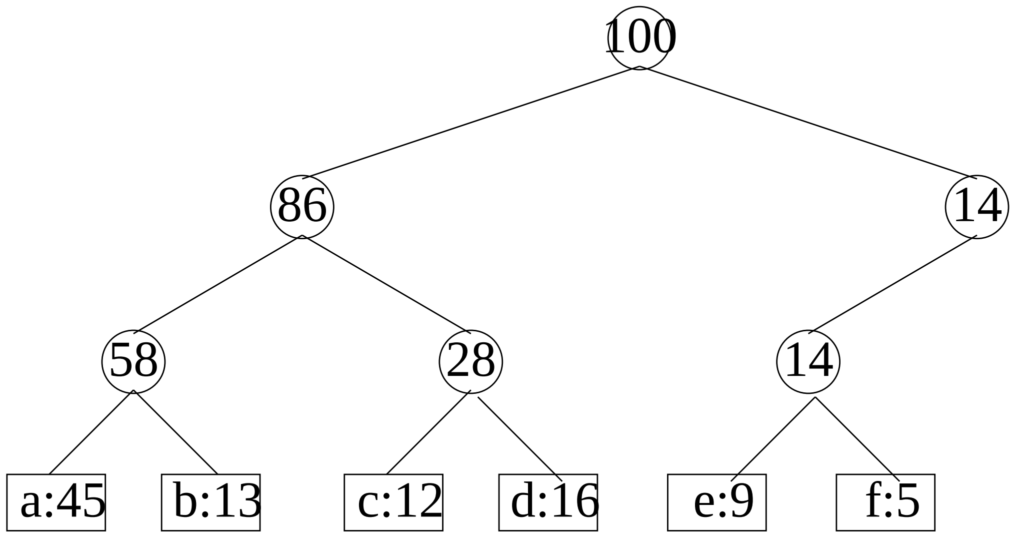
- For fixed length code storage requirement - 3×10^5 bits.
- For variable length code requires 224,000 bits.
  45000+ 69000+36000+48000+36000+20000
- Gives a 25% saving in storage.
- Variable length code uses smaller code for frequently occurring character.

Prefix Code

- Code must have prefix property.
- No code is a prefix of any code word.
- So, decoding 001011101 can be done by prefix each time.
  - First two characters are a·a (0 is not a prefix of any code)
  - Next character must be b (101 is not a prefix of any code)
  - Last character must be e.
- Only constraint is – a representation such that first character can be picked easily from the code.

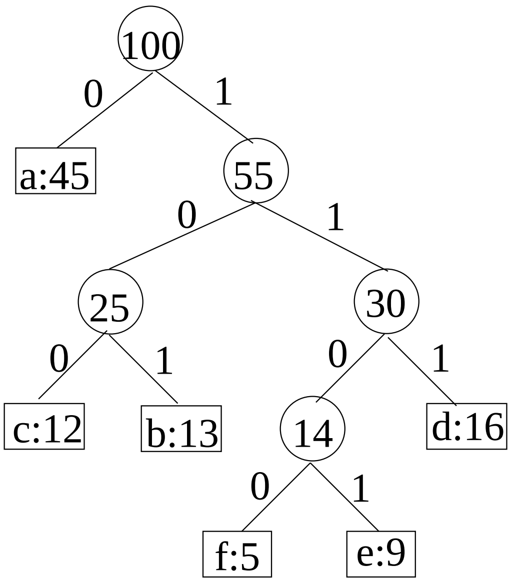
Greedy Algortihm

- A binary tree whose leaves represent character is a convenient representation.
- The code for a character at a leaf is obtained by traversing the tree path from the root to a leaf.
- A left branch appends a 0 and a right branch appends 1 to code.



Variable Length Code

- Representation for variable length code.



The Algorithm

- Let $f(c)$ be frequency of character c in the file.
- Let $d_T(c)$ be depth of the character in the code.
- Then, the number of bits required for encoding

$$B(T) = \sum_{c \in C} f(c) \cdot d_T(c)$$

- Construction of code is done in bottom up manner.

The Algorithm

- Start with $|C|$ nodes. Let c_1, c_2 be characters with least frequencies.
- Merge nodes representing c_1, c_2 forming combined character c' with frequency equal to $f(c_1) + f(c_2)$.
- Replace c_1, c_2 by c' in C.
- Allocate a new node for character c'.
- Let c_1 be left child and c_2 be right child of the c'.
- Repeat the merging process with set C of nodes until $|C|$ becomes 1.

Running Time

- We can form a heap of the character frquencies.
- So finding characters with the smallest and the next smallest frequencies requires 2 `deletemin` operations.
- The frequency of combined character should be inserted after each merge operation.
- Every time one character is reduced. So $|C| - 1$ iterations are needed.
- Building heap initially requires O(n) time.
- Thus total running time is O($n \log n$).

Correctness

- For correctness – show that problem of finding optimal prefix code exhibits `greedy choice` and `optimal substructure` properties.
- We first prove the following lemma.

  Let x and y be two characters in C with lowest frequencies. Then there exists an optimal prefix code for C in which codes for x and y differ only in the last bit.

- That is the tree T representing any arbitrary optimal prefix code can be converted to a tree T' representing another optimal prefix code where x and y are siblings.

Correctness

- Let a, b are two characters with maximum depth in T.
- Then a and b are sibling leaves of maximum depth in T.
- WLOG assume that $f(a) \leq f(b)$ and $f(x) \leq f(y)$.
- Since $f(a)$, $f(b)$ are two arbitrary and $f(x)$, $f(y)$ two lowest frequencies in order, $f(x) \leq f(a)$ and $f(y) \leq f(b)$, we can exchanges x with a, then y with b respectively as shown below.

