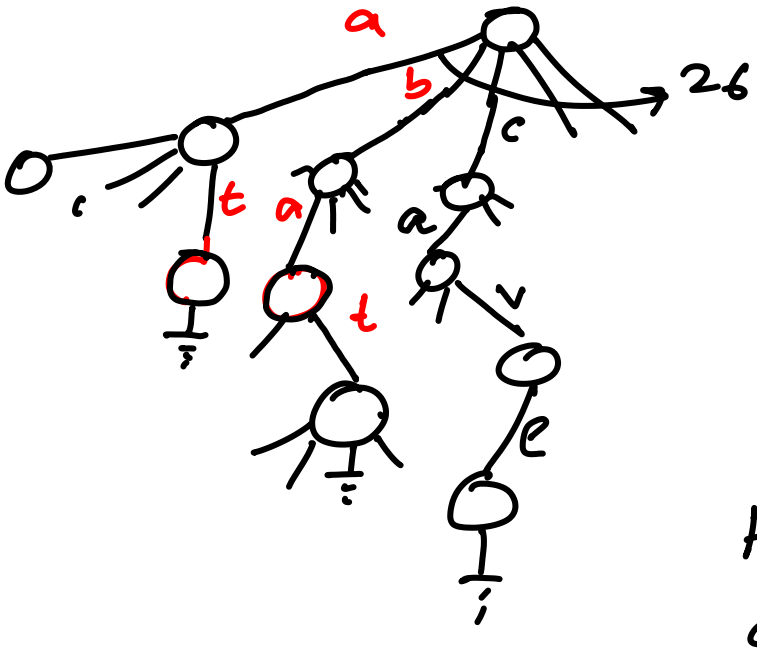
Tries to represent strings
(digital trees)

Σ : finite alphabet

cave, bat, at

A tree to store the set of strings and each node has arity = $|\Sigma|$



String matching,
Common substrings
Longest common prefix

How efficiently can you construct Tries

In particular $O\left(\sum l_i\right) = O(N)$

For $|\Sigma| \leq N$, we could sort in optimal time.

How about space : $|\Sigma| N$

Knapsack problem

Given a knapsack of capacity B and n items with profits p_i $1 \le i \le n$ and weights w_i $1 \le i \le n$ we want to fill up the knapsack so as to maximize profits of the items filled.

Let $x_i = \begin{cases} 1 & \text{if item } i \text{ is included} \\ 0 & \text{otherwise} \end{cases}$

Maximise $\sum x_i p_i$

s.t.

$$\sum x_i w_i \le B$$

$$x_i \in \{0, 1\}$$

Eg. $B = 15$ $n = 4$

p_i	10	10	12	18
w_i	2	4	6	9
x_i	5	2.5	2	2

Greedy approach

(Residual capacity, items to be considered, current profit)

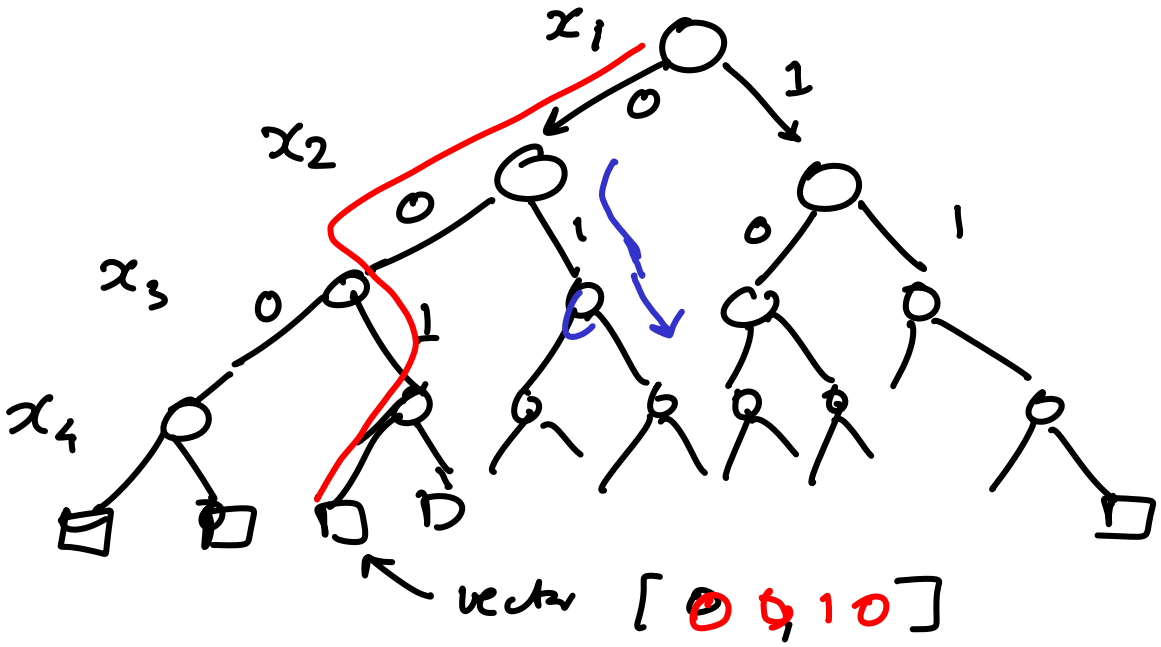
$(15, \{1,2,3,4\}, 0) \longrightarrow (6, \{1,2,3\}, 18)$

$\longrightarrow (0, \{1,2\}, 30)$

Straightforward greedy doesn't get us the best soln for some instances

Even using greedy with ratio will not necessarily work

If we are willing to reconsider past decisions?



- with backtracking, we can obtain the exact soln: however we may visit all leaf nodes $\Rightarrow$ brute force 2^n algo.
- Can we prune the search:
  can we avoid visiting all nodes and still be convinced about the final result?

Can we "estimate" the profit possible by visiting a subtree? ↘ upper bound

In particular an upper bound $U(v)$ ↗ node

At any node v of the search tree, we have a lower bound $L(v)$ and an upper bound $U(v)$

$L(v)$ ↙ the best value we have seen so far

$U(v)$ ↙ the best value we can do by visiting v

If $L(v) > U(v)$ there is no use exploring subtree in v

Even if we were to use pruning strategy, there is no provable guarantee that we will not visit all the 2^n nodes.

A* algorithm

Greedy works for Minimal Spanning Tree