§9.2 : 11 May 2023 MAT 2250

The fundamental theorem of Search Theory

Basic question: Is there an (n,q) tree with n leaves $z_1,\dots,z_n$ with given lengths $\ell(z_1),\dots,\ell(z_n)$, $\ell_i = \ell(z_i)$?

Thm 9.3 (Kraft's inequality: existence of (n,q) tree with prescribed lengths of the leaves)

① Let T be an (n,q)-tree, whose n leaves have lengths $\ell_1,\dots,\ell_n$. Then

$$S(T) := \sum_{i=1}^{n} q^{-\ell_i} \le 1.$$

② Conversely, let $\ell_1,\dots,\ell_n$ satisfy $\sum_{i=1}^{n} q^{-\ell_i} \le 1$.

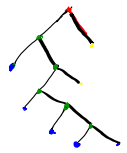
Then $\exists$ an (n,q) tree whose leaves have precisely lengths $\ell_1,\dots,\ell_n$.

③ Rmk: If T is a complete (n,q) tree then $S(T) = 1$.

Proof:

①&③: Let T be an (n,q) tree. If T is not complete, then we can add more vertices (or leaves) to make it complete T'. (T' is (n',q)-tree, $n' \ge n$).

Example $(5,2)$-tree



T' = T + 2 more yellow dots (new leaves)

T' = complete $(7,2)$-tree

T: · blue dots are leaves
· Green dots are internal vertices (except the root, which is red)

$S(T') = S(T) + \sum_{i:\text{yellow dots}} q^{-\ell_i} > S(T)$

So it suffices to show that if T is a complete (n,q)-tree then $(q>2)$

$$S(T) = 1.$$

Proof by induction.

$n = 0$ (trivial tree, with only a root)

$\Rightarrow \ell = 0$: $S(T) = q^{-0} = 1$. (T complete (n,q))

Assume now it is is true for trees with $n \le n_0$.

Now show that it is also true for complete (n,q)-trees T with $n = n_0+1$.

What we do is to transform T into a complete $(n-q+1,q)$ tree T' so that $S(T) = S(T')$.

Because T is a complete (n,q) tree:
$\Rightarrow$ There must exist q leaves with the same immediate predecessor. (So they have the same length, if say ℓ).



We replace this by 1 leaf v^* then get a complete $(n-q+1, q)$-tree T' & length of v^* is $\ell-1$.
(Basically, just delete the q leaves $z_1,\dots,z_q$. Now v is a leaf in the new tree, we call it v^*)

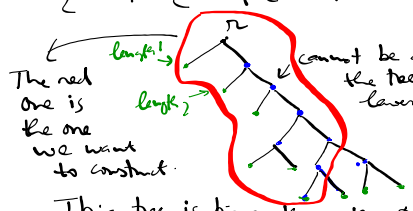
$$S(T') - S(T) = -\sum_{i=1,\dots,q} q^{-\ell} + q^{-(\ell-1)}$$
$i=1,\dots,q$: the q leaves $z_1,\dots,z_q$; $q^{-(\ell-1)}$: the leaf v^*

$$= 0.$$

(2) Assume that $\ell_1,\dots,\ell_n$ satisfy $\sum_{i=1}^{n} q^{-\ell_i} \le 1$.
We will construct by induction an (n,q)-tree whose leaves have lengths $\ell_1,\dots,\ell_n$.

Start from the root.
Then add the leaves of length 1 (meaning the ℓ_i with $\ell_i = 1$)
Now the number of these leaves $\le q$.
$\sum_{i=1,\dots,k} q^{-1} \le 1 \Rightarrow k \le q$.
Add $q-k$ internal vertices which are immediate successors of the root.
Then we can do again, for each of these immediate successor.
Can check that the condition $\sum q^{-\ell_i} \le 1$ gives us room to do this. □
Do an example to see how this goes.

Example 9.4 in the book: (Construct a complete tree, then delete what you don't need)
$n=6$, $q=2$, $\ell_1 = 1$, $\ell_2 = 2$, $\ell_3 = 3$, $\ell_4 = \ell_5 = 5$, $\ell_6 = 6$.
Check $2^{-1} + 2^{-2} + 2^{-3} + 2^{-5} + 2^{-5} + 2^{-6} \le 1$.



The red one is the one we want to construct.
cannot be a leaf, otherwise the tree cannot have leaves of length 5 & 6
This tree is bigger than the tree we want. (it has 4 leaves of length 6). We just choose the subtree which is red then which we want.

Next question:

Q: In average, what is the height of an (n,q)-tree with leaves having depths $l_1,\dots,l_n$ with probabilities $p_1,\dots,p_n$? (More precise later).

Comment about average v.s. worst case:

For an algorithm we may want to say in the worst case, how complex it is. However, this worst case is not representative.

We can have 2 algorithms A_1, A_2 and A_1 in the worst case is worse than A_2 in the worst case. However, in practices, A_1 is usually better than A_2. Two possible reasons:

For a random choice of problem, A_1 can be is usually better than A_2. So we should we estimate the average complexity instead of the worst case complexity.

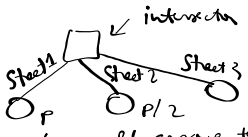
Another reason is the worst case estimate only is valid when n is extremely large, while for practical problems we see, (→ ∞) the size of n is bounded (even maybe quite big.)

An example is algorithms to solve Linear Programming. (There are polynomial algorithms which are not feasible in practice)

Tree with ~~proper~~ probability:

An (n,q)-tree T with leaves $z_1,\dots,z_n$ & probabilities $p_1,\dots,p_n$. $\forall p_i \geq 0$, $\sum p_i = 1$

Example:



The police could arrange that people will go to Street 1 more often than to Street 2, & cannot go to Street 3.

$$\bar{L}(T) = \sum_{i=1}^{n} p_i l_i$$ ← length with probability.

Q: Let T be an (n,q)-tree with leaves of lengths $l_1,\dots,l_n$ & probabilities $p_1,\dots,p_n$. What is a good estimate for $\bar{L}(T)$?

~~Example: if we have uniform probability~~ ~~$p_i = \frac{1}{n}$ $\forall i$~~ ~~then by Theorem 9.1.1.~~

~~$\bar{L}(T) \geq \log_q n$~~

$\bar{L}(T)$ (Shannon)

Thm 9.6: Fix (n,q), ~~$l_1,\dots,l_n$~~, $p_1,\dots,p_n$

$$-\sum_{i=1}^{n} p_i \log_q p_i \leq \min_{T:\,(l_1,\dots,l_n),\,(p_1,\dots,p_n),\,(n,q)} \bar{L}(T) \leq \left(-\sum p_i \log_q p_i\right) + 1$$

↓ entropy!!

Proof:

First, we prove $\bar{L} \geq -\sum_{i=1}^{n} p_i \log_q p_i$.

Put $s_i = q^{-l_i}$

$y_i = p_i$

Then $\sum s_i \le 1$ (Kraft's inequality, part 1 of Thm 9.3)

$\sum y_i = 1$

cross-entropy

$\Rightarrow \sum_{i=1}^{n} y_i \log_q \frac{y_i}{s_i} \ge 0$ (*)

(look for proof in Lemma 9.5)

$\sum_{i=1}^{n} p_i \log_q (p_i \cdot q^{l_i}) \ge 0$

$\sum_{i=1}^{n} p_i (\log_q p_i + l_i) \ge 0$

$-\sum_{i=1}^{n} p_i \log_q p_i \le \sum_{i=1}^{n} p_i l_i = \bar{L}(T)$

Now we show that

$$\bar{L} \le -\sum_{i=1}^{n} p_i \log_q p_i + 1.$$

To this end, we will construct (with help of part 2 of Thm 9.3) an (n,q)-tree with leaves of lengths $l_1, \dots, l_n$ so that

$$\sum_{i=1}^{n} p_i l_i \le -\sum_{i=1}^{n} p_i \log_q p_i + 1.$$

l_i must be a non-negative integer, & how would we choose l_i? We look at (*), then equality happens if $s_i = 1 \cdot y_i \ \forall i$.

The idea is to find l_i so that q^{-l_i} is as close to p_i as possible.

So we choose l_i to be the unique integer satisfying

$$-\log_q p_i \le l_i < -\log_q p_i + 1.$$

Then:

$$\sum_{i=1}^{n} q^{-l_i} \le \sum_{i=1}^{n} q^{\log_q p_i} = \sum_{i=1}^{n} p_i = 1.$$

$\Rightarrow$ By Part 2 of Thm 9.3, there is an (n,q) tree T whose leaves have lengths $l_1, \dots, l_n$.

$$\bar{L}(T) = \sum_{i=1}^{n} p_i l_i < \sum_{i=1}^{n} p_i (-\log_q p_i + 1) = -\sum_{i=1}^{n} p_i \log_q p_i + 1. \square$$

Next question: How do we construct a tree T so that $\bar{L}(T) =$ [illegible] in Thm 9.6.

Huffman's algorithm: Order $p_1, \dots, p_n$ so that $p_1 \ge p_2 \ge \dots \ge p_n$.

If $q-1$ is not a divisor of $n-1$, then add some more leaves with probability 0 so that $q-1$ is a divisor of $n'-1$.

(We want to construct a complete (n,q) tree then delete the leaves with probability 0.)

Initial step: Start with the n leaves & the probabilities.

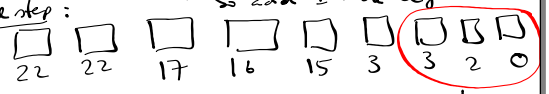
Contracting to the trivial tree:

While not get the trivial tree do:
Choose q leaves with smallest probabilities
Replace them by one leaf, whose probability is the sum of these q leaves.

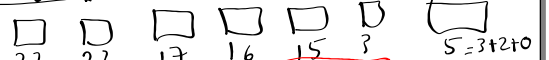
Developing the trivial tree to the tree we want:
Start from the trivial tree with probability 1.
Then gradually undo the contracting step to get the tree.
Example to see how.

Example 9.7: $n = 8$, $q = 3$
$p_1 = p_2 = \frac{22}{100}$, $p_3 = 17$,
$p_4 = 16$, $p_5 = 15$, $p_6 = p_7 = 3$,
$p_8 = 2$.
$8 - 1 = 7$, $3 - 1 = 2$ not a divisor of 7. So add 1 more leaf with prob 0.

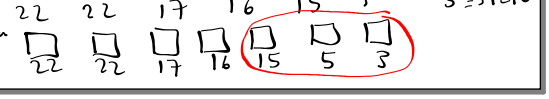
Initial step:

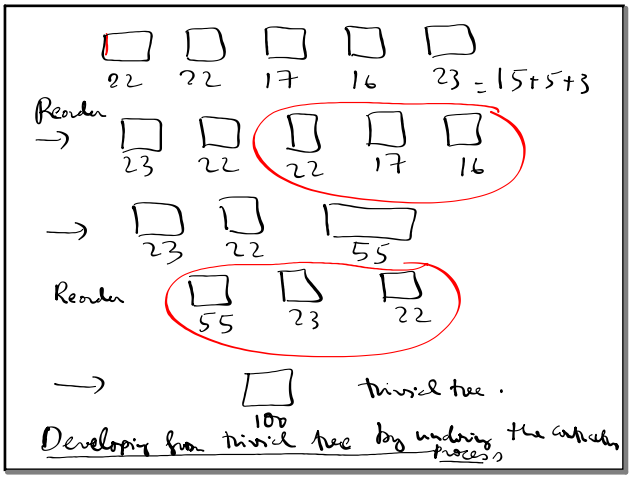


Contracting step:

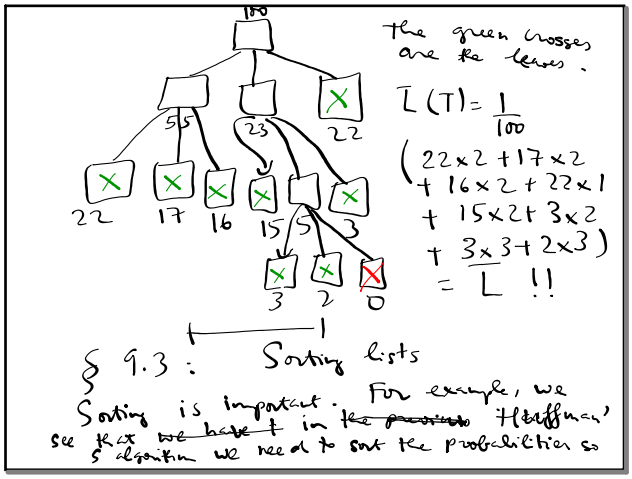


Reorder





Developing from trivial tree by undoing the contracting process



§ 9.3: Sorting lists

Sorting is important. For example, we see that we have to in the ~~process~~ Huffman's algorithm we need to sort the probabilities so

that $p_1 \geq p_2 \geq \cdots \geq p_n$.

Q: Assume that we have a list of n different numbers $a_1, \ldots, a_n$. Find a way to permute them in to $b_1, \ldots, b_n$ so that $b_1 < \cdots < b_n$.

Rmk: This is stated for numbers but of course we can do for anything which has an order.

For example, for the English vocabulary, we could give an order $A < B < C < \cdots$ & hence we can define orders for words, for example Admin < Beginner (lexicographical order).

General approach: We want to construct a tree T, each vertex will involve comparing 2 numbers a_i & a_j, and each internal vertex has 2 children immediate successors depending on $a_i < a_j$ or $a_i > a_j$.

The leaves will be the $n!$ possibilities of permutations of $a_1, \ldots, a_n$.

So: we have a $(n!, \leq 2)$-tree.

By Thm 9.1:

$$L(T) \geq \lceil \log_2 n! \rceil$$

So the complexity of ~~this problem~~ any algorithm for this problem is at least $\lceil \log_2 n! \rceil$.

We will learn about algorithms whose complexity is $O(\lceil \log_2 n! \rceil)$, optimal up to some constant.