Feb 7, 2011

Design of Algorithms by Induction

The celebrity Problem:

A nice example in Algorithm design whose solution does not require scanning all the data (or even a significant part of it).

A celebrity among n persons is someone who is known by everyone but does not know anyone. The problem is to identify the celebrity, if one exists, by asking question only of the form "Excuse me, do you know the person over there?" (we assume all answers are correct and even celebrity answers).

The goal is to minimize the number of questions. Since there are $\binom{n}{2} = \frac{n(n-1)}{2}$ pairs we can have $n(n-1)$ question maximum.

The problem: Given an $n \times n$ matrix whose ij entry is 1 if the ith person knows the jth person, and 0 otherwise, determine whether there exists an i such that all the enteries in the ith column (except for the iith entry) are 1, and all entries in the ith row (except for iith entry) are 0.

Again the idea is to use induction: base case with two persons is easy. Say we know the solution for $n-1$ persons, what about for n.

Note that since there is at most one celebrity three possibilities we have 1) the celebrity is among the first $n-1$ 2) the celebrity is the nth person 3) there is no celebrity.

Case 1) is easy just to check the nth person knows the celebrity and the celebrity does not know the nth person.

The other two cases are harder to determine the nth person is the celebrity, we may need to ask $2(n-1)$ question. then we can have $n(n-1)$ questions that we tried to avoid.

The trick is to solve the problem "backward". It may be hard to identify a celebrity, but it is easier to identify someone as a non-celebrity.

If we eliminate someone, then the size of the problem is reduced from n to $n-1$.

the Algorithm is as follows: we ask Alice whether she knows Bob.
If Alice says no, Bob is <u>not</u> a celebrity and we can eliminate him
If Alice says yes, Alice is not a celebrity and we can eliminate her.
In either case, we eliminate one person. we the find (by induction) a celebrity among n-1 persons. If there is no celebrity, the algorithm terminates; otherwise we check that A knows the celebrity and that the celebrity does not know A.
eliminated guy
we have at most 3(n-1) questions: n-1 questions in the first phase to eliminate n-1 persons, and then at most 2(n-1) questions to verify that the candidate is indeed a celebrity. (instead of n(n-1) questions in the worst case).
*Note that here our instance of n-1 person is <u>smart</u> and not <u>arbitrary</u> (see the code of the Alg in the book).

Evaluating Polynomials:
The problem: Given a sequence of real numbers $a_n, a_{n-1}, \dots, a_1, a_0$, and a real number x, compute the value of the polynomial $P_n(x) = a_n x^n + a_{n-1}x^{n-1} + \dots + a_1 x + a_0$.
Induction can lead to a very good solution:
The first natural attemp is to reduce the problem by removing a_n and thus we are left with evaluating polynomial
$P_{n-1}(x) = a_{n-1}x^{n-1} + a_{n-2}x^{n-2} + \dots + a_1 x + a_0$
<u>IH:</u> we know how to evaluate a polynomial represented by the input $a_{n-1}, \dots, a_1, a_0$ at the point x (i.e. we can compute $P_{n-1}(x)$).
Now $P_n(x) = P_{n-1}(x) + a_n x^n$ (i.e. compute x^n, multiply it by a_n and add the result to $P_{n-1}(x)$
Here we need $n + (n-1) + \dots 1 = \frac{n(n+1)}{2}$ multiplications and n additions.
(here the power of x is computed from scratch.
<u>stronger IH</u>: we know how to compute the value of the $P_{n-1}(x)$ and we know how to compute x^{n-1}. Then we need one multiplication for x^n one for $a_n x^n$ and one addition at the end (2n multi. and n additions total)
What about if we remove a_0 instead of a_n.

Define $P'_{n-1}(x) = a_n x^{n-1} + a_{n-1} x^{n-2} + \cdots + a_1$

IH (reversed order) since we remove a_0 instead of a_n: We know how to evaluate $P'_{n-1}(x)$ with coefficients $a_n, a_{n-1}, \ldots a_1$.

Then $P_n(x) = x P'_{n-1}(x) + a_0$ we need one multiplication and one addition on

(n multi. and n additions total)

Algorithm Horner's Rule

Input: $\bar{a}: a_0, a_2, \ldots a_n$ (coefficients of a polynomial), and x (a real number)

Output: P (the value of the polynomial at x).

```
begin
  $P := a_n$;
  for $i=1$ to $n$ do
    $P := x * P + a_{n-i}$
end.
```

Finally Finding the maximum consequtive subsequence:

<u>The problem</u>: Given a sequence $x_1, x_2, \ldots x_n$ of real numbers (not necessarily posit

find a subsequence $x_i, x_{i+1}, \ldots x_j$ (of consequtive elements) such that the sum of the numbers in it is maximum over all subsequences of consecutive element

For $2, -3, 1.5, -1, 3, -2, -3, 3$ we have maximum subsequence $1.5, -1, 3$ with sum 3.5.

If all the numbers are negative, then the max seq is empty with sum 0.

The straight-forward IH: We know how to find the maximum subsequence in sequence of size $< n$.

For $n=1$, the max subseq. consists of the single number if it is non-negative or empty otherwise.

Consider $S = (x_1, x_2, \ldots x_n)$ of size $n>1$. We know the max subseq in $S' = (x_1, x_2, \ldots, x_{n-1})$. If max subseq is empty in S', then every number in S' is negative and we only need to consider x_n. Say the max subseq in S' is $S'_M = (x_i, x_{i+1}, \ldots x_j)$ for $1 \le i \le j \le n-1$ if $j = n-1$ (namely the max subseq. is a <u>suffix</u>), we only need to check x_n for extension.

However if $j<n-1$, then there are two possiblities. Either S'_M remains maximum, or there is another subsequence, which is not maximum in S', but is maximum is S when x_n is added to it.

The idea is to <u>strengthen the induction</u> by knowing only the max subseq is not enough. so

stronger IH: we know how to find, in sequence of size $<n$, a maximum subseq. overall, and the max subseq that is a suffix.

Now that we know both, it is much easier. We add x_n to the max suffix. If the sum is more than the global max subseq., then we update it (as well as the new suffix) - otherwise we retain the previous max subseq. <u>We are not done yet.</u> We also need to find the new max suffix. Not always adding x_n is good. if it is negative, we take the empty set as max suffix.

so with cn operations, we find the max subseq. instead of $\binom{n}{2}n$ operations if we tried all i, j. $\boxed{\text{See the code of the algorithm in the book}}$

<u>Strengthening the Induction Hypothesis</u>

Instead of $P(<n) \Rightarrow P(n)$, we can add another assumption Q which makes the proof easier $[P \text{ and } Q](<n) \Rightarrow P(n)$. However this is the common mistake, we should prove $[P \text{ and } Q](<n) \Rightarrow [P \text{ and } Q](n)$.

It is cruicial to follow the IH precisely.

See Common Error is section 5.11