Brief solution outlines for the problems

Problem A - The Mailbox Manufacturers Problem

Composed in Linköping, Sweden.

Task

The problem was to find the minimum number of crackers required to find exactly how many crackers a certain mailbox prototype could withstand, given the number of available mailboxes and the use of an optimal strategy.

Solution

This is a classic application of the concept known as *dynamic programming*. In that technique you express the solution to a problem in terms of solutions to the same problem of smaller size. Then you solve the original problem by solving the smaller problems first, which in their turn may be solved by considering even smaller problems. The trick is to find the recurrence equation, and in this particular problem it is indeed rather tough. Let $T_{i,j,k}$ denote the number of crackers required in the worst case to solve the problem if you have i mailboxes, and you know that j crackers are not enough to destroy a mailbox, but $k+1$ crackers will. As pointed out in the problem text, given a single mailbox which you know will not blow up with j crackers, you have no other alternative than testing, in order, if it holds for $j+1$,$j+2$,... crackers. Thus,

$$T_{1,j,k} = \frac{k(k+1)}{2} - \frac{j(j+1)}{2}$$

Now, what if you have i mailboxes and you know they resist j crackers but not $k+1$, what is the first number of crackers you should try on the first mailbox? Let's say you try l crackers and the mailbox get wasted, then you are left with $i-1$ mailboxes, but you know that the mailbox resists j crackers but not l. On the other hand, if the mailbox persists the explosion of the l crackers, you have all i mailboxes intact and you know that a mailbox resists l crackers but not $k+1$. So, the worst case in the best strategy must be

$$T_{i,j,k} = \min_{i<l\leq k} (l + \max(T_{i-1,j,l-1}, T_{i,l,k}))$$

By looping from smaller to larger values, you may easily calculate $T_{i,j,k}$ in the range asked for. A last difficulty is that in the problem you do not know an

upper bound on how many crackers are required to destroy a mailbox which fits m crackers. However, assuming that it cannot withstand $m+1$ crackers (if they would fit) cannot help your strategy, so $T_{i,0,m}$ is the correct answer for i mailboxes.

Problem B - Parallel

Taken from ACM World Finals 1991, testdata composed in Oslo, Norway. It's also availble from Valladolid: `http://acm.uva.es/p/v2/210.html`.

Task

The problem was to simulate several concurrent programs on a single processor machine with a simple mechanism for guarding critical sections.

Solution

It is pretty straightforward what must be done, but there are a few implementation issues to deal with. You have to implement two first-in-first-out queues to keep track of whose turn it is to run. Also, you need a shared databank containing the current values of all the variables 'a'..'z', and a global clock measuring the runtime of the current program. This is a typical example of a "simple" problem with a lot of details which must be remembered.

Problem C - The Piano Tuners

Composed in Lund, Sweden.

Task

The problem was to find out if a list of scheduled piano moves could be carried out by a few men within the time they had previously promised, preferably by avoiding working on the weekends.

Solution

There are several approaches to solve this problem. The simplest is perhaps acting after the heuristic principle "the most urgent jobs first". The reason why this works is best illuminated by a contradiction argument. Suppose you have to choose between two piano moves which you may do today, but one of them is more urgent than the other, and the optimal strategy would be to move the second piano today, and the more urgent some later day d. Then, since the time intervals must overlap in the region from today to day d, you may exchange the jobs for each other, without increasing the amount of required work! Consequently, you may safely assume that in the optimal strategy, urgent

jobs are handled first. Simulating the simple heuristic scheduling is an easy task, once for work at weekdays only, and once for all days of the week.

Problem D - Collecting Beepers

Composed in Bergen, Norway.

Task

The problem was to find the shortest tour for a robot to pass by a few beepers which should be collected and then return to the original position.

Solution

Perhaps the most well-known NP-complete problem, the 2-dimensional geometric traveling salesman problem (TSP), here in Manhattan metric. There *are* more efficient solutions than testing all permutations of the beepers and investigate how long the tour gets, but there is no need for them in this case. There can only be at most 10 beepers, and thus at most $10! = 3628800$ tours to test. Enumerating the tours is easily accomplished by a recursive procedure. One should of course stop investigating a tour as soon as it gets longer than the best found so far. Adding this simple if-statement typically improve on the running time drastically, since a lot of tour starts by visiting the same beepers in the same order.

Problem E - Quantum

Composed in Linköping, Sweden.

Task

The problem was to investigate if it is possible to transform a binary number of fixed length to another using a few bit-fiddling operations, and if so, to find the cost of the cheapest transformation.

Solution

This is an example of the problem known as "single-source-shortest-path". Imagine the different binary words as vertices in a graph and connect two vertices u, v with an arc from u to v if there is an operation transforming u to v. Let the arc have the weight of the operation cost. The problem is perhaps best solved with the famous algorithm named after Dijkstra, using a Heap data structure as priority queue. Solutions using naïve priority queues may fail to finish within the time limit!

Problem F - The Goldbach Conjecture

Taken from Asian Regional 1998, testdata created in Oslo, Norway. It's also availble from Valladolid: `http://acm.uva.es/p/v6/686.html` and `http://acm.uva.es/p/v5/543.html`.

Task

The problem was to find the number of ways an even positive integer greater than 2 could be expressed as the sum of two prime numbers.

Solution

Yet another famous problem. The simplest solution is to test all pairs of integers who sum up to the given even integer, if they are both primes. This strategy may possibly take too much time though. A more sophisticated approach, is to calculate and tabulate which numbers in the given range are primes, using the Erastothanes sieve algorithm. This algorithm is described in most mathematical encyclopedias.

Problem G - Downpayment

Composed in Umeå, Sweden.

Task

The problem was to find the best way to borrow money to afford bying a house, given the interest rates of several credit institutes.

Solution

Once again, a problem which is best solved using dynamic programming. Let $M_{i,j}$ denote the least amount of money your resulting loan could be after j elapsed months, if you finish a contract with the ith credit institute this month. This choice make it somewhat awkward to start the dynamic programming, since in the start of a new loan, you may choose any credit institute for free. However, the recurrence equation for the following months are easier.

Problem H - Simplified λ-Calculus

Composed in Trondheim, Norway.

Task

The problem was to evaluate a single-variable λ-expression.

Solution

The tricky part in this problem is to keep track of which variables are bound, and which are not. The standard solution is to implement a recursive descent parser, so that all subexpression get their own little stack. Then find the most nested function and evaluate it.