GB21802 - Programming Challenges

Week 2 - Problem Solving Paradigms (Search)

Claus Aranha
caranha@cs.tsukuba.ac.jp

College of Information Science

2015-05-6,9

Last updated May 6, 2016

Last Week Results

Week 1

Problems Solved

- Division Of Nlogonia: 28/28
- Cancer Or Scorpio: 24/27
- 3n+1: 25/25
- Request for Proposal: 19/22

Manaba Submission: 29 Students

Week 2

Problems Solved

- Jolly Jumpers: 26/26
- Army Buddies: 15/21
- Rotated Square: 17/17
- File Fragmentation: 4/5
- Contest Scoreboard: 10/11
- Multitasking: 3/5
- Jolibee Tournament: 7/7

Manaba Submission: 25 students

Don't Give Up!

Special Notes

If the UVA Judge is offline

Sometimes the UVA Judge gets overloaded. If this happens near the deadline, send your programs to MANABA before the UVA deadline. Re-send your programs to UVA as soon as it becomes available again.

If uDebug result is different from UVA result

uDebug is a volunteer service. Although it is very useful, its correctness is not guaranteed. If you see a result that indicates that uDebug is wrong, contact the maintainers of that site, they will be grateful for the report.

Always try to submit programs early!

Quick Review of Last Week

- Linear Arrays (unidimensional and bidimensional) are KISS data structures;
  - A lot of problems can be solved by iterating on arrays
  - Always allocate a bit more than you need.
- Bitmasks are quick and efficient ways to store sets of True/False values;
- Tree data structures can search very efficiently
  - Instead of implementing your own, learn to use the structures available from the library.

Topic for this week: Search

What is Search

In day to day life, we say we are searching for something when we are trying to find where this something is located.

- Keys of your bycicle;
- Your wallet;
- Your cellphone;

When searching, we think of our Goal (the thing we are searching), and the search space (the number of places where the thing could be hidden)

The thing you search is always in the last place you look

(*By definition!*)

What is a "Search Problem"?

We call a problem a search problem, if we can describe the problem as checking multiple *answers* in order to find one or more *solutions*.

- Answers to a search problem could be correct or incorrect;
- Answers can also possibly have higher or lower scores;
- We can sort the answers by score, or by some other criteria;

Many problems in programming challenges can be described as search problems! (even if sometimes there are better ways to describe them)

Sample Search Problems

Request For Proposal (Week 1)

You are given a set of proposals with the number of satisfied requirements, and the total cost. Find the proposal with the highest number of requirements, and the lowest total cost.

Search Space: The set of proposals in the input.

File Fragmentation (Week 2)

You are given a set of binary fragments. Find a binary value that match all existing fragments.

Search Space: The set of all binary values with size $< n$

Now you are thinking with search spaces

When we define a problem as a search problem, we can use strategies based on organizing the search space, and systematically examining each solution contained in it.

Questions about the algorithm:

- How is the search space represented?
- How are the solutions ordered?
- In what order are the solutions tested?
- How many solutions are tested?

Thinking with search spaces: Request for Proposal

- How is the search space represented?
  For each proposal, the number of requirements and the cost are stored.
- How are the solutions ordered?
  The array of proposals is ordered by highest number of requirements and lowest cost.
- In what order are the solutions tested?
  Highest item in the array of proposals are checked first.
- How many solutions are tested?
  Only the first item is returned.

Of course, for many problems there is more than one way to fill this pattern.

Search Paradigms

These are some common approaches for search problems:

- Complete Search/Brute Force;
- Divide and Conquer;
- Greedy Approach;
- Dynamic Programming (Next week!)

As we said before, some problems can use multiple approaches (but not all of them are equally good!)

Theoretical Example (1)

Search Space

You have an array A of n integers ($n < 10K$), where the value of each integer a_i is ($0 \leq a_i \leq 100K$).

Imagine the following problems:

- Find the Largest and the smallest element of A;
- Find the k^{th} smallest element of A;
- Find the largest gap G such that $x, y \in A$ and $G = |x - y|$;
- Find the longest increasing subsequence of A;

Theoretical Example (2)

How costly would be to search the solutions for each of the four problems described?

- Find the Largest and smallest element of A: O(n) - single pass, and we cannot really go faster than this.
- Find the k^{th} smallest elements of A:
  - Repeat the search k times: $O(n^2)$ in the worst case;
  - Order the number and search: $O(n\log n)$
- Fing the largest gap:
  - Try all possible pairs: $O(n^2)$
  - Greedy: Find the smallest and largest numbers $O(n)$ (you have to prove this works)
- Longest increasing subsequence:
  - Test all possible subsequences (brute force): $O(2^n)$
  - Dynamic programming: $O(n^2)$
  - Greedy search: $O(n\log k)$ – can you prove this?

Complete Search/Brute Force (1)

Complete Search algorithms are expected to test all (or almost all) solutions.

Complete Search are usually called "Brute Force". But because they are often the best way to solve a problem, we use a nicer name here.

Complete Search/Brute Force (2)

Structure of a Complete Search:

- Test all existing solutions
  Usually achieved through either for loops or recursive calls;
- Prune, Prune, Prune
  Remove bad solutions (or bad sets of solutions) as you go, by "breaking" early form loops, or setting good ending conditions to the recursive calls.

Complete Search Example: UVA 725 – Division

Problem Summary

Given an integer N, find all pairs of numbers *abcde* and *fghij* so that $fghij / abcde = N$ and all 10 digits are different.

Example: $N = 62$

79546 / 01283 = 62
94736 / 01528 = 62

Consider this problem for a bit before I show how to solve it using search.

Complete Search Example: UVA 725 – Division

Full Search Solution:

A naive way to solve the problem is to test all $0 \leq x \leq 99999$, calculate $y = x * n$, and test whether x and y have all different digits.

```
for (int x = 0; x < 99999; x++)
{
  y = x*n;
  digits = test(x,y);
  if (digits == 1<<10 - 1) printf("%0.5d/%0.5d=%d\n",y,x,N);
}

int digits(int x, int y)
{
  int used = (x < 10000);
  int tmp;
  tmp = x; while (tmp) {used |= 1 << (tmp%10); tmp /= 10; }
  tmp = y; while (tmp) {used |= 1 << (tmp%10); tmp /= 10; }
  return used;
}
```

Complete Search Example: UVA 725 – Division

Pruning the complete loop:

- What is the absolute minimum and maximum for x? 01234:98765
- Maximum for Y is also 98765, so the actual maximum for x is $x < 98766/n$
- Can we cut the digits test earlier?

Considerations about complete search

- A bug-free complete search should ALWAYS be correct.
  - A complete search tests all solutions, so it should always find the correct one;
  - Of course, in many cases, checking all solutions takes too long;
- Complete Search should always be solution considered (KISS principle)
  - If the problem is so small that a better solution is overkill;
  - If you are running out of ideas, or take too many WAs;
  - Prune, prune, prune!
- Sometimes, you can use a simple complete search on a hard problem to get an idea of what sort of result is expected.
  - Use it to generate solutions for test cases in problems that generate TLEs.

Complete Search Example 2: Simple Equations

Problem Summary – UVA 11565

Find x, y, z so that $x + y + z = A$, $x * y * z = B$, $x^2 + y^2 + C^2 = C$, $1 \leq A, B, C \leq 10000$.

We need to test sets of x,y,z, but how do we set the limits for these values?

Example 2: Simple Equations – initial pruning

Consider $x^2 + y^2 + x^2 = C$. Since $C \leq 10000$, the maximum range for x, y, z must be $-100, 100$.

Therefore, here is the Complete Search Loop

```
bool sol = false; int x,y,z;
for (x = -100; x <= 100 && !sol; x++)
  for (y = -100; y <= 100 && !sol; y++)
    for (z = -100; z <= 100 && !sol; z++)
      if (y != x && z != x && z != y &&
          x + y + z == A && x * y * z == B && x*x + y*y + z*z =
            if (!sol) printf("%d %d %d\n", x,y,z);
            sol = true;
          }
```

Can you think of other ways to prune the loop?

Example 2: Simple Equations – more pruning

There are many other ways that we can prune the loop:

- We can change the range using the actual input values of A, B, C
- We only need one solution. We can break the loop once we find it.
- We can consider the other two equations, specially equation 2.

This week's problem: "Simple Equations – Extreme!" has a much higher range for A, B, C. You need a lot of pruning to avoid a TLE!

Complete Search: TIPS 1

Divide and Conquer

D&C is a problem-solving paradigm in which a problem is made simpler by 'dividing' it into smaller parts.

- Divide the original problem into sub-problems;
- Find (sub)-solutions for each sub-problems;
- Combine sub-solutions to get a complete solution;

Examples

Quick Sort, Binary Search, etc...

Greedy

Greedy Example 1

Coin Change:
Given a target value V and a list of coin sizes, what is the minimum number of coins that we must use to represent V?
Example: V = 42, Coins = 25, 10, 5, 1 (a One coin means we can always make any value)
However, if V = 6, and coins = 4,3,1, the greedy algorithm does not reach an optimal solution.

Greedy Example 2

Load balancing: (UVA 410)
You have C chambers, and S < 2C specimens. You need to decide where each specimen should go to minimize "imbalance".
Insights:
- A chamber with 1 individual is always better than a chamber with 0 individuals.
- Order of chambers does not matter.
Greedy algorithm: Order the individuals by weight, and put one in each chambers until the chambers are full, then add one in each chamber backwards.

Greedy Example 3 - Dragon of LooWater (11292)

List of knights (with heights) and dragons (with diameter). Find the minimal height of knights that cut the heads of all dragons (knights can only cut the heads of dragons when D <= H).
Sort knights and dragons, and from smallest to biggest dragon, match with the smallest knight possible to solve.

Problem Discussion

- Division
- Social Constraints
- Simple Equations - EXTREME!!
- Bars
- Rat Attack
- Little Bishops
- Water Gate Management
- Through the Desert
- Dragon of Loowater
- Shoemaker's Problem

Search Algorithms are an important area of research

Heuristics