Han Yu a0077864M

1. **What I learnt**

3.2 Complete Search

Filtering when complete search like exercise 3.2.1.1. It is very smart to loop to 98765/n.

Stop the search when it already breaks the rules.

Do some pre-calculation is good for query. Just spend memory space or time first time.

Sometime change a side to solve problem is a good way like UVA 10360. Count cell is worse than count bomb.

3.3 Divide and Conquer

Binary search the answer. There is one way to search the double type answer. We can first set the while condition (hi>lo+1). The then multiple hi and lo by 10, also times 10 for all the other points. Then use the while loop again while(hi>lo+1) then the integer part is hi/10, decimal part is hi%10.

Example uva 11516 part of the code:

while ( hi - lo > 1 ) {

int mid = (lo + hi) / 2;

if ( check (mid \* 2) ) hi = mid;

else lo = mid;

}

lo \*= 10;

hi \*= 10;

for ( int i = 0; i < m; i++ )

houseNo [i] \*= 10;

while ( hi - lo > 1 ) {

int mid = (lo + hi) / 2;

if ( check (mid \* 2) ) hi = mid;

else lo = mid;

}

printf ("%d.%d\n", hi / 10, hi % 10);

3.4 Greedy

Load balancing.

Interval covering problem: It is important to sort in greedy method.

1. **Do some exercise**

3.2.1.1: It can avoid divide operation, so only integer will be appeared in calculation. Also the loop is quite small which can be from 01234 to 98765/n.

3.2.1.2: It should be AC since 10! = 3628800 about 3.5M. < few 10s of Million

3.2.1.3: Mentioned in 3.2.1.1

3.2.1.4: If (row[i]==b&&i!=a) continue;

Do UVA10360 Rat Attack

Complete UVA11935

Solve UVA11413

* + - 1. Solve Uva410 by greedy: nice load balancing

1. **What is inside but not clear**

None ^\_^

But a problem about Uva410: actually it is easy to calculate the imbalance, but it is hard to out the result in certain order like the sample output(appear order). What is lucky is though the order is different to the simple output of my program, but still got AC…

1. **What is not inside but I can add**

One difference between Greedy and DP:

For example: Knapsack problem. If it is a 0-1 Knapsack, only take the item or not take, you cannot take half or one third, then it should be solved by DP , Greedy doesn`t work. But if you can take partly, then Greedy method can used to find the most effective item. Maybe this can only be used for some condition, but I think can add an example to compare Greedy and DP.

How to binary search the double type answer:

Just above in What I learnt 3.3 Divide and Conquer