Member Count: 672,342 - August 26, 2014 [Get Time]

Hello, **faijur** | Logout

My TopCoder

Competitions
Overview
Copilot Opportunities
Design
Development
UI Development
QA and Maintenance
CloudSpokes
Algorithm
High School
The Digital Run
Submit & Review

TopCoder Networks
Events
Statistics
Tutorials
Forums
Surveys
My TopCoder
Help Center
About TopCoder

Member Search:
Handle: Go
Advanced Search

Dashboard > TopCoder Competitions > ... > Algorithm Problem Set Analysis > SRM 605

Search

TopCoder Competitions

SRM 605

View | Attachments (0) | Info | Browse Space

Added by [[rng_58]] , last edited by vexorian on Jan 27, 2014 (view change)
Labels: (None) EDIT

Archive
Match Overview
Discuss this match

Single Round Match 605

Tuesday, January 21st, 2014

Match summary

More than 1800 coders participated in SRM 605 , a problem set by **Witaliy**. A match with interesting ad hoc and dynamic programming problems. In division 1, the coders were greeted by a problem that required coders to be crafty in implementation or spend too much time coding. **semiexp** found a way to code some not-so simple code in just few minutes and got the fastest score. The second problem was the interesting dynamic programming one that required care and analysis to be implemented correctly. **yeputons** got the best score in mere 13 minutes. The hard problem was mostly about decomposing a problem into simple properties and was solved by only 6 coders of which, **semiexp**, got the best score (that's two speed records in a single match). Not content with dominating the problem scores, **semiexp** also got 50 challenge points to magnify the lead over second place: **Nerevar**. Also breaking the 1000 points barrier: **VARtem** got the third place.

The Problems

AlienAndPassword | AlienAndGame | AlienAndSetDiv2 | AlienAndHamburgers | AlienAndSetDiv1 | AlienAndPermutation

AlienAndSetDiv1

Rate It | Discuss it

Used as: Division One - Level Two:

Value	450
Submission Rate	119 / 662 (17.98%)
Success Rate	84 / 119 (70.59%)
High Score	**yeputons** for 376.51 points (13 mins 5 secs)
Average Score	242.78 (for 84 correct submissions)

The Division 2 version of the problem is very similar to this one. The only difference is that the division 2 version asks for the difference between $A[i]$ and $B[i]$ to be at most K, whilst the division 1 version wants the difference to be at least K.

This difference makes the problem slightly more complicated, but nothing big.

Good integers

Assume we add integers in decreasing order from $n = 2N$ to $n = 1$. So for $N = 4, K = 2$ we start with $n = 8$. Consider that the sets are sorted, there are two options: we can add n to the right-most position of A or B, pick to add n to A

```
A=???8
B=????
```

Now we should add $n = 7$. Since $K = 2$, we cannot add n to B, because the difference between 7 and 8 is not enough. So we are forced to add it to A:

```
A=??78
B=????
```

We are finally able to add $n = 6$ to B, or we can add it to A if we like.

The same optimizations that were used in the division 2 version come into play. The symmetry makes it so that we can always assume that the set with *unmatched* integers is $\widehat{A}$ and we should ignore all integers that were already matched.

So it is all about the unmatched set $\widehat{A}$. This set will have many elements. This time it can have integers greater than $n + K$. The important observation is, however, that we do not really need to differentiate between those integers that are *at least* $n + K$. Once a integer becomes at least $n + K$, that is all we need to know about it. In other words:

```
A=5678
B=??34
```

And:

```
A=4578
B=??36
```

Should have the same result, because the remaining numbers to add are $n = 1$ and $n = 2$, 4,5 and 6 are all greater than or equal to 4.

From this, we can represent the set of unmatched integers $\widehat{A}$ with two parameters:

- g: The number of *good* integers. Integers that are free to match. Integers that are at least $n + K$.
- s : The set of integers between $n + 1$and $n + K - 1$that are in $\widehat{A}$.

There are at most 2^{K-1} possible values for s and $O(N)$ values for g. This means that there are $O\left(2^{K-1} \cdot N^2\right)$ states. The rest is to simply implement dynamic programming using the same strategy as in the division 2 version.

Code

Using bit masks in c++. This time we should make sure that whenever the new set contains $n + K - 1$, that element should be removed from the set and g incremented.

```
static const int MOD = 1000000007,  MAX_N = 50, MAX_K = 10;
int K;

int dp[2*MAX_N + 1][2*MAX_N + 1][ 1 << (MAX_K-1) ];

long rec(int n, int g, int mask)
{
    long res = dp[n][g][mask];
    if (res == -1) {
        res = 0;
        if (n == 0) {
            res = ( ( (g == 0) && (mask == 0) ) ? 1 : 0 );
        } else if ( (mask == 0) && (g == 0) ) {
            // can put n in either of the sets.
            if (K == 1) {
                // In this case, even this element is going to be "good"
                res = 2 * rec(n - 1, 1, 0);
            } else {
                // Add to the set, it will be element 0
                res = 2 * rec(n - 1, 0, 1);
            }
        } else {
            if (g > 0) {
                // match with a good one
                int nmask = (mask << 1); //shift right
                int ng = g - 1;
                if (nmask & (1 << (K-1) ) ) {
                    // The (K-1)-th element becomes a good one
                    nmask ^= (1 << (K-1) );
                    ng ++;
                }
                res = rec(n - 1, ng, nmask );
            }
```

Alternative solutions and additional comments.

<Place your comments here>

Next problem: AlienAndPermutation

Author

By **vexorian**

TopCoder Member

Editorial feedback	Choose
I liked it.	✓
I didn't like it.	✓

Comments (Hide Comments)

While the editorial is preliminary, you are welcome to suggest changes and corrections. The editorial will be fully editable once the final problem explanation is ready.

When posting a comment thread, make sure to specify the problem you are talking about.

Posted by vexorian at Jan 23, 2014 00:29 | Reply To This

AlienAndSetDiv1: "Once a integer..." should be "Once an integer...".

Posted by johnathan79717 at Jan 23, 2014 01:20 | Reply To This

int the problem AlienAndSetDiv2
int the c++ solution(use map)
is "indent" means nothing(it seems you didnt use it)?
thx~

Posted by ray007great at Jan 23, 2014 12:59 | Reply To This

It was used for debugging and forgot to remove it.

Posted by vexorian at Jan 23, 2014 14:29 | Reply To This

For Division Two - Level One (AlienAndPassword), how can qzpm5n, #1 on this level, complete everything in (0 mins 8 secs)? 8 seconds is not enough for me even to complete reading the question statement. Any magic?

Posted by bjiang78 at Jan 23, 2014 13:42 | Reply To This

One possibility is the use of multiple accounts. Using one account read the problem and get the solution. Using another account compile and submit it.

Posted by sushilpandey at Jan 29, 2014 02:01 | Reply To This

nice solution explaintion!

Posted by ray007great at Jan 23, 2014 23:03 | Reply To This

One liner in c++
int getNumber(string S) {
return std::unique(S.begin(), S.end()) - S.begin();
}

Posted by jitendra_theta at Jan 24, 2014 01:57 | Reply To This

please explain it also + the complexity ...

Posted by kavish_mnnit at Jan 26, 2014 10:01 | Reply To This

AlienAndSetDiv1

can be solved as C(2N, N) - AlienAndSetDiv2(K-1)

🙂

Posted by vlad_d at Jan 24, 2014 13:35 | Reply To This

AlienAndSetDiv2

You say that "The set for n contains numbers from n+1 to n+K , inclusive.", That's not

true right?
Because in your own example, 6 is not present in the set which contains 7 and 8?!

Posted by vishwasrao at Jan 24, 2014 15:25 | Reply To This

Interpret as: May contain only numbers from n+1 to n+K.

Posted by vexorian at Jan 24, 2014 15:31 | Reply To This

AlienAndGame - O(W*H) Solution

```
int getNumber(vector <string> board)
{
int res = 0, dp[2][64][64];
for (int i = 0; i < board.size(); i++)
for (int j = 0; j < boardi.size(); j++)
dp[0]i[j] = dp[1]i[j] = 0;

for (int i = 0; i < board.size(); i++)
if (i == 0)
{
for (int j = 0; j < boardi.size(); j++)

Unknown macro: { if (board[i][j] == 'W') dp[0][i][j] = 1; else dp[0][i][j] = 0; dp[1][i][j] = !dp[0][i][j]; }

}
else
{
for (int j = 0; j < boardi.size(); j++)
{
for (int k1 = 0; k1 < 2; k1++)
{
int a = j==0?0:dp[k1][i-1][j-1],
b = dp[k1][i-1][j],
c = j==0?0:dp[0]i[j-1],
c2 = j==0?0:dp[1]i[j-1];
if (boardi[j] == 'W')

Unknown macro: { dp[0][i][j] = 1 + min(a, min(b, c)); dp[1][i][j] = 0; }

else

Unknown macro: { dp[0][i][j] = 0; dp[1][i][j] = 1 + min(a, min(b, c2)); }

}
}
}

for (int i = 0; i < board.size(); i++)
for (int j = 0; j < boardi.size(); j++)
res = max(res, max(dp[0]i[j], dp[1]i[j]));
return res*res;
}
```

Posted by thebvog at Jan 27, 2014 04:08 | Reply To This

Hi , I was trying to code AlienAndGame which passed on sample test cases but failed on system test cases. please let me know if i have a done a logical mistake.
I will call GetMaxSquare(r,c) in the main function

```
int GetMaxSquare(int i,int j,vector<string> &board)
{
if(i == 0 || j == 0)
return 1;

if((boardi[j] == boardi[j-1]) && (board[i-1][j-1] == board[i-1][j]))

Unknown macro: { int size = GetMaxSquare(i,j-1,board); size= min(min(size,GetMaxSquare(i-1,j,board)),GetMaxSquare(i-1,j-1,board)); maximum =max(maximum,size+1); return size+1; }

else
return 1;
}
finally return maximum * maximum ;
```

Posted by fusionreborn at Jan 28, 2014 14:59 | Reply To This

awesome editorial..first time, I read editorial on topcoder. I learned a lot of things. thanks man 🙂

Posted by anup1pma at Jan 29, 2014 12:20 | Reply To This

o(wh) solution

```
public : int getNumber(vector<string>b)
{
int m=b.size();
int n=b[0].size();
int a[m][n];
for(int i=0;i<m;i++) ai[0]=1;
for(int i=0;i<n;i++) a[0]i=1;
int ans=1;
for(int i=1;i<m;i++)
{
for(int j=1;j<n;j++)
{
if(bi[j]==bi[j-1] && b[i-1][j]==b[i-1][j-1])

Unknown macro: { int temp; if(a[i][j-1]==a[i-1][j]&& a[i][j-1]==a[i-1][j-1])temp=sqrt(a[i][j-1])+1; else temp=sqrt(min(a[i][j-1],min(a[i-1][j-1],a[i-1][j])))+1; a[i][j]=temp*temp; }

else ai[j]=1;
if(ai[j]>=ans) ans=ai[j];
// cout<<i<<"\t"<<j<<"\t"<<ai[j]<<endl;
}
}
return ans;
}
```

Posted by taree.earth at Jan 30, 2014 02:38 | Reply To This

I got a O(K^2 * N) solution: in C++

```
typedef long long ll;

using namespace std;
```

```
template<ll mod = 1000000007, ll maxn = 100>
struct NCM
{
private:
ll fac[maxn + 1];
ll inv[maxn + 1];
ll nCm[(maxn + 1) * (maxn + 1)];
public:
NCM()
{
fac[0] = inv[0] = 1LL;
for (int i = 1; i < maxn; ++i)

Unknown macro: { fac[i] = fac[i - 1] * i % mod; inv[i] = powMod(fac[i], mod - 2); }

for(int i = 1; i <= maxn; ++i)
{
for(int j = 0; j <= maxn; ++j)

Unknown macro: { nCm[i * (maxn + 1) + j] = fac[i] * inv[j] % mod * inv[i - j] % mod; }

}
}

ll operator()(int n, int m)

Unknown macro: { return nCm[n * (maxn + 1) + m]; }

private:
ll powMod(ll a, ll b)
{
ll ret = 1LL;
while (b > 0)

Unknown macro: { if ((b & 1) > 0) ret = ret * a % mod; a = a * a % mod; b >>= 1; }

return ret;
}
};

template<ll mode = 1000000007>
struct Multi
{
public:
ll operator()(ll a, ll b)
{
int ret=0;
for(a%=mode,b%=mode; b > 0; a =(a<<1)%mode,b>>=1)
{
if (b&1)

Unknown macro: { ret = (ret + a)%mode; }

}
return ret;
}
};

class AlienAndSetDiv2
{
public:
int getNumber(int N, int K)
{
```

```
static Multi<> multi;
static NCM<> nCm;

int C[51][51] =

Unknown macro: {0}

; // C[N][K], Counting permutation with |Ai - Bi| equals K (== only, < excluded).
int S[51][51] =

; // S[N][K] = C[N][1] + C[N][2] + ... + C[N][K]
memset(C, 0, 51 * 51 * sizeof(int));
memset(S, 0, 51 * 51 * sizeof(int));

// Think about S[N][K] (N = 1, 2, ... ) with N == K, the permutations with |Ai - Bi| <= K will be either of the below two forms:
// 1 a2 a3 ... N , 1 < a2 < a3 < ... < N
// b1 b2 b3 ... bn , b1 < b2 < ... < bN
// or
// 1 a2 a3 ... an, 1 < a2 < a3 < an
// b1 b2 b3 ... N, b1 < b2 < b3 < bn.
// and please note the set A and B could be switched to each other,
// so there's a formula to calculate the total number of permutations with |Ai - Bi| <= K
// S[N][N] = 2 * (nCm(2 * N - 2, N - 1) + nCm(2 * N - 2, N)).

for(int n = 1; n <= N; ++n)
{
if(n == 1)

Unknown macro: { S[n][n] = 2; }

else

Unknown macro: { S[n][n] = 2 * nCm(2 * n - 2, n - 1) + 2 * nCm(2 * n - 2, n); }

}

for(int n = 1; n <= N; ++n)
{
for(int k = 1; k <= n && k <= K; ++k)
{
// if k == n, calculate C[n][n] using S[n][n] computed above.
// C[n][n] = S[n][n] - S[n][n - 1]

if(k == n)

Unknown macro: { C[n][k] = S[n][n] - S[n][n - 1]; }

else // C[n][k] = C[1][1] * C[n - 1][k] + C[2][2] * C[n - 2][k] + ... + C[k][k] * sum(C[n - k][1], C[n - k][2], ..., C[n - k][k])
{
for(int i = 1; i <= k; ++i)

Unknown macro: { C[n][k] += multi(C[i][i], (i < k ? C[n - i][k] }

S[n][k] = S[n][k - 1] + C[n][k];
}
}
}

return S[N][K];
}
};
```

Posted by wadewu at Feb 04, 2014 15:39 | Reply To This

Add Comment

Home | About TopCoder | Press Room | Contact Us | Privacy | Terms
Developer Center | Corporate Services