Eternal Victory ::

এই প্রবলেমটাতে আসলে যা বলা হইছে আমাদের একটা ট্রি দেওয়া আছে (বিভিন্ন সিটি নিজেদের মধ্যে কনেক্টেড হয়ে ট্রি করছে) । এই ট্রি এর সবগুলো নোডে আমরা মিনিমাম কত cost এ visit করতে পারি । আমরা ১ নাম্বার নোড থেকে যাত্রা স্টার্ট করব এবং যেকোন নোড এই থামতে পারব ।

যেহেতু এইটা ট্রি এর মানে হইল আমাদের কোন লুপ নেই । লুপ না থাকার কারণে আমরা জানি যে যেকোন একটা লিফ নোড এ আমরা থামব এবং এইখানে কতগুলো রাস্তা আমাদের দুইবার যাতায়াত করতে হবে । যেহেতু আমরা সব থেকে মিনিমাম cost বের করছি আমরা চাব ১ নাম্বার নোড থেকে এর সব থেকে দুরের নোডে যেতে যে যে রাস্তা দিয়ে যাইতে হয় তারা যেন একবারই ভিজিট হয় আর বাকি সব নোড হবে দুইবার করে ।

```
const int NX = 1e5 + 10 ;


Long dis[ NX ] ;


vector < int > adj[ NX ] , cost[ NX ] ;
int n ;


void solve()
{


    cin >> n ;
    Long ans = 0 ;
    rep ( i , n - 1)
    {
        int x , y , c ;
        cin >> x >> y >> c ;
        adj[x].pb( y );
        adj[y].pb( x );
        cost[x].pb( c );
        cost[y].pb( c );
        ans += ( c + c );
    }
    For( i , n ) dis[i] = ( 1e12 );
    dis[1] = 0 ;
    priority_queue < pair < Long , int >  > pq ;
    pq.push( mp( 0 ,  1 ) ) ;
    while( !pq.empty() )
    {
        pair < Long , int > now = pq.top();
```

```
            pq.pop();
            int x = now.ss ;


            Long c = now.ff * -1 ;
            int sz = adj[x].size();
            rep( i , sz )
            {
                int u = adj[x][i];
                Long cc = cost[x][i];
                if( dis[u] > dis[x] + cc )
                {
                    dis[u] = dis[x] + cc ;
                    pq.push( mp( -dis[u] , u ) );
                }
            }
        }
        Long mx = 0 ;
        for ( int i = 1 ; i <= n ; i++ ) mx = max( mx , dis[i] );
        ans -= mx ;
        cout << ans << endl ;






    }
```

ReplacingDigit ::
টপ কোডারের এই প্রবলেম এ বলা হয়েছে আমাদের কিছু product এর প্রাইজ ট্যাগ দেওয়া হবে এবং সাথে কিছু এক্সট্রা ডিজিট ও দেওয়া হবে আমারা চাইলে আমাদের কোন প্রোডাক্টের প্রাইজ ট্যাগর কোন ডিজিটকে বদলাতে পারি কিন্ত্ আমাদের কোন এক্সট্রা ডিজিট আমরা আমাদের প্রোডাক্টের প্রাইজ ট্যাগর সাথে যোগ করতে পারব না ।আমাদেরকে বলতে হবে হাইস্ট কত আমরা এই প্রোডাক্টগুলো থেকে পেতে পারি । লিমিট এ একটা জিনিস বলা আছে হাইস্ট 10^6 প্রোডাক্টের প্রাইজ হতে পারে ।
যদি ধরে নেই আমরা গ্রিডি সল্ভ করব তাহলে গ্রিডি সলুশ্যন এর সব থেকে গুরুত্বপূর্ণ পয়েন্ট হচ্ছে current stage এর highest possible gain । কি কি ফ্যাক্টর তা আমাদের দিবে এইটা আমাদের বের করতে হবে ।

১। most right digit থেকে আমরা দেখব আমরা কোন নাম্বার চ্যাঞ্জ করে বেশি লাভ করতে পারি কিনা ।
২। যদি পারি তাহলে সবথেকে smallest most right digit কে আমরা all possible extra digit থেকে highest possible value digit দ্বারা replace করব যা আমাদের highest profit ensure করবে ।

৩। একই প্রসেস আমরা সব ডিজিট এর জন্য ব্যাবহার করব ।

এখন আমাদের এই ব্যাপারগুলাকে একসাথে merge করে answer বের করতে হবে ।

```
class ReplacingDigit
{
public:
    int getMaximumStockWorth(vector <int>, vector <int>);
};
int mul[ 10 ];
vector < int > digit[ 10 ];


int ReplacingDigit::getMaximumStockWorth(vector <int> A, vector <int> D)
{


    int m = 1 ;
    for( int i = 0 ; i < 8 ; i++ )
    {
        mul[i] = m ;
        m *= 10 ;
    }
    int sz = A.size();
    for( int x = 0 ; x < sz ; x++ )
    {
        int num = A[x];
        int idx = 0 ;
        while( num > 0 )
        {
            digit[idx++].pb( num % 10 );
            num /= 10 ;
        }
    }
    int total = 0 ;
    int idx = 8 , add = 9 ;
    for( int i = 7 ; i >= 0 ; i-- )
    {
        sz = digit[i].size();
        if( sz == 0 ) continue ;
        sort( digit[i].begin() , digit[i].end());
        while( idx >= 0 && D[idx] == 0 )
        {
```

```
            idx--;
            add--;
        }


        for( int j = 0 ; j < sz ; j++ )
        {
            if( idx == -1 || digit[i][j] >= add )
            {
                total += ( mul[i] * digit[i][j]);
            }
            else
            {
                total += ( mul[i] * add );
                D[idx] -= 1 ;
                while( idx >= 0 && D[idx] == 0 )
                {
                    idx--;
                    add--;
                }


            }
        }




    }
    return total;
}
```

view raw **greedy_3_2.cpp** hosted with ❤ by **GitHub**

To Add or Not to Add::

কোডফরসেস এর এই প্রবলেম এ বলা হয়েছে আমাদের একটা N সংখ্যক number এর array দেওয়া আছে এবং সাথে একটা নাম্বার K ও দেওয়া আছে । K এর কাজ হল , আমরা K সংখ্যকবার চাইলে array এর কিছু element (যদি খালি একটাও হয়) ১ করে বাড়াতে পারি । আমাদেরকে এই array থেকে highest frequency এর নাম্বার প্রিন্ট করতে হবে , যদি একাধিক থাকে তাওলে সবচেয়ে ছোট নাম্বার প্রিন্ট করতে হবে ।

এই প্রবলেমটা অনেকভাবে সল্ভ করা যেতে পারে এর মধ্যে two pointer technique use করে আমরা খুব সহজে করতে পারি (two pointer technique কি এইটা জানা থাকলে এই ব্লগটা আগে পড়ে আসতে হবে) । এরপর range এর মধ্যে commutative sum এর মাধ্যমে কোন নাম্বার এর জন্য হাইস্ট কতবার নাম্বারটা হতে পারে

তা আমরা বের করতে পারি ।

প্রথমে আমরা sort করে নিব array টাকে , sort করাটা important কারন আমরা এর উপর range উপর কোন নাম্বার নিয়ে তা কতটা বানানো সম্ভব query করব , যা শুধু মাত্র একটা নাম্বার বানানোর জন্য এর সমান এবং সামান্য ছোট নাম্বারগুলাকে এর সমান বানাতে minimum কতটা adding দরকার হবে যা sorting sequence থেকেই আমরা পাই ।

```
const int NX = 1e6 + 10 ;




Long inp[ NX ] , sum[ NX ] ;


int n , m ;






int main()
{
   // I will always use scanf and printf
   // May be i won't be a good programmer but i will be a good human being
     n = II , m = II ;
     rep ( i , n )
     {
         inp[ i ] = LL ;
     }
     sort ( inp , inp + n );


     Long freq = -1 , ans , tmp ;
     int p = 0 ;
     rep ( i , n )
     {
         sum [ i + 1 ] = inp[ i ] + sum [ i ] ;
         while( (( inp[i] * (Long) ( i - p + 1 ) ) - ( sum [ i + 1 ] - sum[ p ] ) )> m )
p++;
         if( freq == -1 || freq < ( i - p + 1 ) )
         {
```

```
               ans = inp[ i ] ;
               freq = i - p + 1 ;
           }
       }
       cout << freq << " " << ans << endl ;


      return 0;
  }
```

two pointer এর মাধ্যমে প্রবলেম সল্ভ এর জন্য আমাদের দুইটা range পয়েন্ট থাকে , starting point and end point . আমরা যেহেতু iterative করে array এর প্রতিটি ভ্যালু নিয়ে দেখছি আমাদের উপরের কোডের 'i' হচ্ছে end point এবং 'p' হচ্ছে আমাদের starting পয়েন্ট । এখন p পয়েন্ট থেকে i পয়েন্ট পর্যন্ত যে নাম্বারগুলা আছে তাদের যদি আমরা 'i' পয়েন্ট এ যে নাম্বারটার সমান করতে চাই তাহলে যতবার adding operation লাগবে তা K(কোডে m) এর ছোট বা সমান হচ্ছে কিনা তা 26 number লাইনের while loop দিয়ে চেক করা হচ্ছে , পরে answer update করা যায় কিনা check করা হচ্ছে ।

উপরের কোডের একটা মাত্র সাইন চ্যাঞ্জ করলে তা minimum number থেকে maximum number পাওয়া যাবে highest frequency এর জন্য , কোন সাইনটা চ্যাঞ্জ করতে হবে তা পাঠকদের খুঁজে বের করার জন্য ছেড়ে দিলাম ।

এই লিখাটা আর বড় করছি না , আমার মনে হয় প্রোগ্রামিং কনটেস্ট এ আগ্রহ থাকার পরও প্রোগ্রামিং কনটেস্ট এ অনেক এর ভাল করা সম্ভব হয় না quality problem solve না করার কারনে । ১০০টা same logic এর প্রবলেম সল্ভ করে যে লাভ টা হয় না , তার চেয়ে ৫টা different logic এর প্রবলেম সল্ভ করে অনেক কিছু জানা হয় । এখন একজনের পক্ষে হয়তো এই ৫টা different problem এর আইডি জানা কস্টকর , যে কাজটা একজন কোচ হয়তো করে দেন বা কোন সিনিয়র বড় ভাই ভার্সিটির যিনি নিজে অনেক টাইম দিছেন এইসব এর পিছনে । ভাল প্রবলেম এর আইডি মনে রাখা এমন মাঝে মধ্যেই এই প্রবলেম গুলার সল্যুশন লজিক দেখা important . যেইটা আমি নিজে অনেক পরে বুঝেছি বলে এখনও মাঝে মধ্যে আফসোস হয় । সব ভার্সিটির ট্রেনিং প্রসেস এক হওয়া সম্ভব না , এখন আমারা যারা নিজেরা সাফার করছি বা করিও নেই (অনেক ভাল যারা করেছেন এবং করছেন) তাদের দায়িত্ব হল এই প্রবলেম গুলার আইডিয়া বলে দেওয়া , সম্ভব হলে কিছু সল্ভ লজিক সহ যেন অন্য যে কেউ তা থেকে নিজে নিজে একটা শিখতে পারে । চীনা এবং রাশিয়ানরা কেন প্রোগ্রামিং কনটেস্ট এ এত ভাল করে তা একটা ভাল কারন ট্রেনিং ম্যাটেরিয়াল নিজেদের ভাষায় অনেক বেশি তাদের , যেইটা এখনও বাংলাতে নেই । যদি আমি গত ৪ বছর যাবত অনেক চ্যাঞ্জ দেখছি , আমি যখন স্টার্ট করি এক ফাহিম ভাইয়া এর ব্লগ বাদে কিছু ছিল না , পরে সাফায়াত ভাইয়া এখন প্রায় ব্যাসিক সব কিছু নিয়ে লিখে ফেলেছেন কিন্তু এখনো প্রবলেম ম্যাটেরিয়াল নিয়ে সবাই আগ্রহী না । আমার ব্লগের মাধ্যমে কারো কোন উপকার হইতে এইটা অনুরোধ থাকবে আপনিও বাংলাতে প্রবলেম সল্ভিং নিয়ে লিখা শুরু করুন । ইনশাল্লাহ আইওআই এবং এসিএম এর গোল্ড খুব একটা বেশি দূরে নয় :)