প্রথমেই আসা যাক , **greedy** কি ? **greedy** হল ভবিষ্যতের এর কথা চিন্তা না করে বর্তমান অবস্থা গুলা বিবেচনা করে বেস্ট একশনটা নেওয়া । হয়ত এইটা পরবর্তীতে সবথেকে **optimal** নাও হতে পারে । greedy solution তো optimal না তাহলে কেনই বা আমি greedy solution নিতে চাব । প্রথমত greedy solution time efficient । এমন অনেক ক্ষেত্রেই ধরে নেওয়া হয় greedy solution টাই **best possible Ans** . greedy solution যেহেতু implement করা সহজ তাই অনেক optimized problem এর solution এর জন্য greedy use করা হয় ।

কিছু পরিচিত greedy process Change Making , kruskal Algorithm , Activity Selection .

Change Making :

Change Making problem এ বলা হয় আমার কাছে অনেক গুলা বিভিন্ন মানের মুদ্রা আছে । আমাকে কোন সব থেকে কম মুদ্রা ব্যবহার করে Change দিতে হবে । আমি কিভাবে কাজটা করব ।

এর প্রসেস হচ্ছে আমি সবসময় সবথেকে বড় মুদ্রাটা থেকে স্টার্ট করব এবং যতক্ষণ পর্যন্ত না এর ভ্যালু আমার change (একটা নিয়ে Total change ভ্যালু থেকে subtract করা তো আছেই) এর amount থেকে বড় হয়ে যাবে আমি নিতে থাকব , যদি তা বড় হয়ে cross হয়ে যায় তাহলে এর পরের value দিয়ে কাজ করার চেস্টা করতে থাকব ।

Code টা কিছুটা এমন

```
Say coin[] = { 100 , 50 , 20 , 10 , 5 } ;
Total_change --> amount we need to change
Ans <-- 0
for i = 1 to total coin // Descending order sorted , large to small
while( Total_change >= coin[i] )
{
            Ans++ ;
            Total_change -= coin[i];
 }
print --> Ans ;
```

আবারও বলে থাকা ভাল coin change এর জন্য greedy solution optimal না , optimal হল dp   solution . কিন্তু টাইম লিমিট অনেক সময় dp solution এর থেকে greedy solution   টাকেই optimal ধরে নেওয়া হয় । যদি এমন দেখা যায় coin গুলোকে ascending order এ সর্ট করার পর 2*coin[i] <= coin[i+1] তাহলে দেখা যাবে greedy solution best optimal result এই দিচ্ছি ।

Activity Selection Problem :

Activity selection problem টা এমন আমাকে অনেক গুলা কাজ দেওয়া আছে । start time ও end time সহ । আমি একটা সময় শুধু মাত্র একটা কাজ এই করতে পারি । আমাকে যদি Nটা কাজ দেওয়া হয় তাহলে আমি সব চেয়ে বেশী কয়টা কাজ করতে পারব । এইখানে overlapping possible না মানে একটা কাজ শেষ না করে কোন কাজ শুরু করতে পারব না । এই প্রবলেম এর solutionটা অনেক সুন্দর । আমি কাজগুলাকে তাদের end time এর বেসিস এ sort করব । এর পর আমি যে কাজটা সবার আগে আসবে তা করব । এমন এর পর এ সেই কাজটা শুরু করব যার start time এই কাজের end time এর থেকে বেশী ।

Codeটা কিছুটা এমন

```
struct habijabi
    {
               int start_time , end_time ;
     } work[ MX ] ; // work structure
    bool cmp ( habijabi A , habijabi B)
     {
               if( A.end_time == B.end_time ) return A.start_time < B.start_time ; // if
both end time is equal then start time earlier work will have better position
             return A.end_time < B.end_time ;
      }
      sort( work , work+total , cmp ) ; // end time basis sort
     int Ans = 0 , prev_end = -1 ; // always small for 1st slot
     for ( i = 0 ; i < total ; i++ )
    {
         if( work[i].start_time > prev_end )  // we can take it
           {
            Ans <- Ans+1 ;
            prev_end = work[i].end_time
            }
    }
    print -- > Ans
```

view raw **greedy2.cpp** hosted with ❤ by **GitHub**

এখন দেখা যাক এইভাবে করলে আমি কেন সব সময় বেস্ট Ans পাচ্ছি । আমি যদি end time ধরে সর্ট করে
কাজ স্টার্ট এর জন্য নেই আমি সবসময় সেইসব কাজ এই নিব যাদের end time অন্য কাজগুলা থেকে আগে   শেষ হচ্ছে   মানে আমি best option পাচ্ছি আরো বেশী কাজ স্টার্ট করার ।

Active selection problem থেকে বুঝা যায় আসলে greedy কেন আসলে মাঝে মধ্যে dp থেকে ভাল । ধরুন আমার N টা কাজ এর লিস্ট আছে যাদের থেকে আমার বেস্ট লিস্ট করতে হবে যাতে আমি সবথেকে বেশী কাজ শেষ করতে পারি । DP এর জন্য আমার possible option 2^N . এখন N এর মান যদি অনেক বড় হয় তাহলে তা strict time limit জন্য খুব একটা ভাল উপায় না । আমি TLE খাব অনেক code এই । তাই মাঝে মধ্যে greedy is good .

Interval scheduling problem :

Interval scheduling (Greedy) problem এ আমাকে অনেক গুলা কাজ এর start এবং end time দেওয়া হইছে । আমাকে প্রতিটা কাজ এর জন্য একটা program assign করতে হবে । আমাকে বলতে হবে কিভাবে করলে সব থেকে কম program assign করতে হবে । এইখানে overlapping possible না মানে একটা কাজ শেষ না করে কোন প্রোগ্রাম ফ্রী হবে না । মানে ৬ মিনিট এ যদি কোন কাজ শেষ হয় আর অন্য একটা কাজ ৬ মিনিট থেকে start হয় তাহলে ৬ মিনিট এর কাজ না শেষ করে যেহেতু অন্য কাজ শুরু করা যাবে না তাই এইখানে দুইটা প্রোগ্রাম লাগবে । আমি এইখানে sort করব । sort করার সময় end point , start point কে আলাদা ভাবে mark করব । start point priority পাবে মানে sorting এ সেম পয়েন্ট এ end , start থাকলে start আগে থাকবে ।

```cpp
struct abc
{
    int value , mark ; // mark 0 for start point , 1 for end
} Inp [ Mx + Mx ] ;
bool cmp ( abc A  , abc B )
{
    if ( A.value == B.value ) return A.mark < B.mark ;  // start mark age thakbe
    return A.value < B.value ;
}


int main()
{
    int n , i  , x , y , idx = 0;
    cin >> n ;
    for ( i = 0 ; i < n ; i++ )
    {
        cin >> x >> y ;
        Inp[idx].value = x ;
        Inp[idx++].mark = 0 ;
        Inp[idx].value = y ;
        Inp[idx++].mark = 1 ;
    }
    sort(Inp , Inp+idx , cmp );
    int Ans = -Inf ;
    int cur = 0 ; // eita count korbe koyta program ekhon run korche
    for ( i = 0 ; i < idx ; i++ )
    {
        if( Inp[i].mark == 0 ) // mane notun program start hoiche
            cur++;
        else cur-- ; // program off hoiche
        Ans = max(Ans , cur );
    }
    print -- > Ans ;
    return 0 ;


}
```

কোডিং এ আমি চেক করব current position maximum কয়টা program স্টার্ট আছে , এইটাই আমার Ans . কারন এই সময় এই আমার সব থেকে প্রোগ্রাম রান করে রাখতে হবে । এর চেয়ে কম নিলেও আমার হবে না বেশী নিলে এক্সট্রা প্রোগ্রামগুলা বসে থাকবে ।

Minimum Spanning Tree (kruskal) :

minimum spanning tree ও greedy problem এর জন্য ভাল উদারন । শাফায়াত ভাইয়া অনেক ভাল টিউটরিয়াল লিখছে kruskal . আশা রাখি এইখান থেকে একটু দেখলেই সবার clear হয়ে যাবে ।

কিভাবে আইডিয়া পাব এইটা greedy solution হতে পারে ? যেকোন প্রবলেম এর solution idea পাবার পূর্বশর্ত হল এইরকম প্রবলেম অনেক সল্‌ভ করা । আমি যদি দেখি Ans গুলা current best option থেকে আসসে তাহলে আমি greedy solution এর কথা ভাবতে পারি । অনেক greedy problem এর সর্টিং , বাইনারি সার্চ এর দরকার হয় মানে সর্টিং , বাইনারি সার্চ করে বেস্ট পসিবল উত্তর পাওয়া যায় । এইসব ব্যাপার মাথায় রাখতে হবে । অনেক Dp প্রবলেম টাইম লিমিট এর মধ্যে করার জন্য code optimized করার প্রয়োজন হয় । তখন অনেক কেস greedy process থেকে বাদ দেওয়া হয় । তাছাড়া কোন প্রবলেম dp ,আর কোনটা greedy তার মধ্যে difference করার জন্যও আমাদের greedy ভাবনা ভাবতে হবে । greedy মানে আমি নরমাল এই বেস্ট পসিবলের জন্য যা চিন্তা করি (human brain current stage থেকে একটা certain stage পর্যন্ত ভ্যালু ভাবতে পারে , তাই আমাদের চিন্তার ধরন greedy) . Uva আর Light Oj তে অনেক প্রবলেম আছে greedy এর জন্য । এইগুলা কিছু করলেই আরোও idea clear হবে সবার ।
problem
Uva Problem
Light Oj

সবাইকে অনেক শুভ কামনা :)