DP on Tree

Andreas Cendranata
Binus

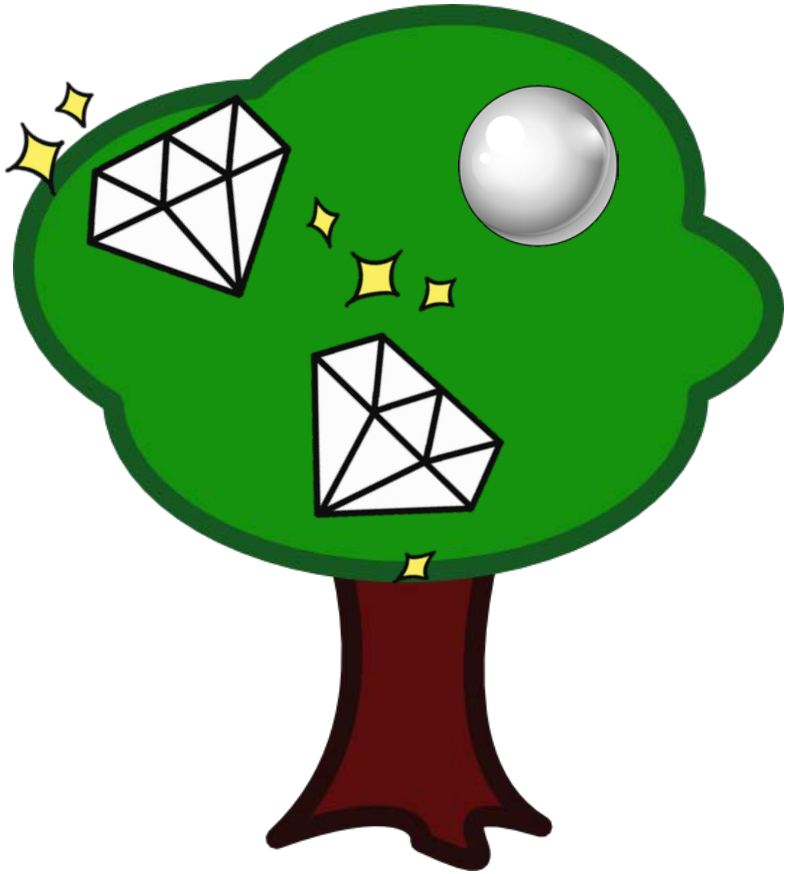
DP on Tree

Apa itu DP on Tree??

DP on Tree

Apa itu DP on Tree??

Diamond Pearl on Tree!!!!



DP on Tree

Apa itu DP on Tree??

DP yang ada di tree :D
(You don’t say)

DP on Tree

Apa itu DP on Tree??

DP yang ada di tree :D
(You don't say)

DP biasa, cuma dilakukan di tree saja
Why tree?? Karena tree itu graf special asiklik yang enak

Yang ada di slide-slide berikutnya, treenya sudah **rooted** yaa :D

Maximum Independent Set

A maximum independent set is an independent set of largest possible size for a given graph G. This size is called the independence number of G, and denoted a(G).[2] The problem of finding such a set is called the maximum independent set problem and is an NP-hard optimization problem. As such, it is unlikely that there exists an efficient algorithm for finding a maximum independent set of a graph. - Wikipedia

Maximum Independent Set

Di general graph, cari maximum independent set harus 2^n di optimize

Jadi kalau nodenya 100000, kompleksitasnya 2^{100000}...

Maximum Independent Set

Di general graph, cari maximum independent set harus 2^n di optimize

Jadi kalau nodenya 100000, kompleksitasnya 2^{100000}...

Selesai program di 10^{30000} tahun!!

LETS USE TIME MACHINE!

Maximum Independent Set

Nah kalau di tree bisa O(N) !

Maximum Independent Set on Tree

Disinilah DP bekerja!

State : [now][status]
now => posisi dimana
status => posisi now sudah diambil atau belum

Maximum Independent Set on Tree

Disinilah DP bekerja!

State : [now][status]
now => posisi dimana
status => posisi now sudah diambil atau tidak

Transisi :
Kalau status sudah diambil, anak-anaknya tidak boleh diambil
Kalau status belum diambil, anak-anaknya boleh diambil, boleh tidak

Maximum Independent Set on Tree

```
3   //Hitung posisi sekarang jika status = 1(diambil)
4
5   int ret = status;
6
7   //Jika status sudah diambil, anak-anaknya tidak boleh diambil
8   if(status)
9   {
10      for(int nex : child[now])
11      {
12          ret += dp(nex, 0);
13      }
14  }
15  else // Jika status belum diambil, anak-anaknya bebas
16  {
17      for(int nex : child[now])
18      {
19          ret += max(dp(nex, 0) , dp(nex, 1));
20      }
21  }
22  return ret;
```

DP on Tree

Selain maximum independent set, ada minimum vertex cover….
Minimum vertex cover adalah himpunan vertex terkecil sehingga untuk setiap edgenya, setidaknya salah satu dari edgenya ada dalam himpunan tersebut.

Seperti maximum independent set, minimum vertex cover di general graph adalah NP-Hard

Kalau di tree??

DP on Tree - Minimum Vertex Cover

State : [now][status]

now => posisi sekarang dimana

status => posisi now sudah diambil atau tidak

Sama dengan maximum independent set...

Transisinya??

DP on Tree - Minimum Vertex Cover

State : [now][status]
now => posisi sekarang dimana
status => posisi now sudah diambil atau tidak

Transisi :
Kalau status sudah diambil, anak-anaknya bebas
Kalau status belum diambil, anak-anaknya **harus** diambil

DP on Tree Example

Source : KTH Challenge 2013

Diberikan sebuah track berbentuk tree dengan N buah node. Setiap node diberi nomor dari 1-N. Akan ada banyak pelari yang akan memakai track ini dengan cara pergi dari node 1 dan akan kembali ke 1 lagi. Track yang setiap pelari lalui akan selalu tepat berjarak S. Disetiap node dapat dipasangi lampu. Karena pelari akan berlari di malam hari, maka setiap edge yang mereka lewati harus minimal salah satu ujungnya ada lampu. Diberikan L buah lampu yang sudah terpasang. Berapa minimum banyaknya lampu tambahan untuk memenuhi syarat tersebut? (Setiap pelari dapat berputar balik meski ditengah-tengah edge)

DP on Tree Example

Source : KTH Challenge 2013

Observasi :
- Hanya pakai edge yang dapat dikunjungi pelari
- Minimum Vertex Cover saja :D

Di slide sebelumnya, minimum vertex cover sudah kan....
Hati-hati dengan lampu yang sudah terpasang supaya tidak double counting...

DP on Tree Example

Diberikan sebuah binary tree dengan N node. Tiap node bernilai V_i
Mau ambil paling banyak K node dimana setiap node yang dipilih merupakan independent set. Nilai dari K node yang dipilih adalah jumlah dari setiap nilai pada node tersebut. Berapa nilai maksimum yang dapat diperoleh?

$1 <= K <= N <= 100$
$1 <= V_i <= 10000$

DP on Tree Example

State : [now][status]**[K]**

K disini sebagai penanda di subtree ini harus ambil K node

DP on Tree Example

State : [now][status]**[K]**

K disini sebagai penanda di subtree ini harus ambil K node

Base case

Di leaf, jika K != status artinya gagal...

DP on Tree Example

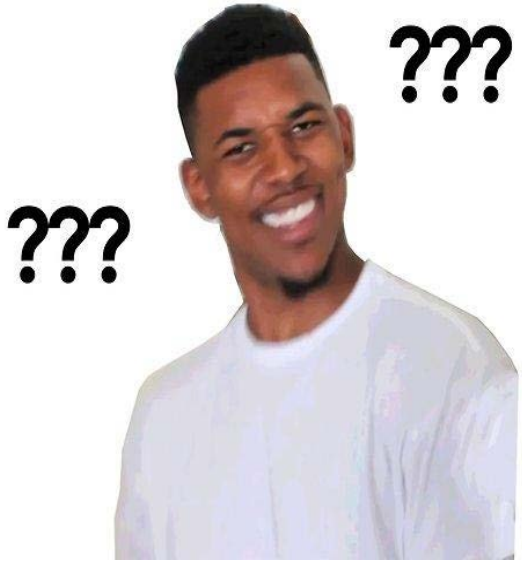
State : [now][status][K]
K disini sebagai penanda di subtree ini harus ambil K node

Base case
Di leaf, jika K beda dengan status artinya gagal…

Transisi :
Jika posisi sekarang diambil, assign ke kiri dan kanan mau ambil berapa tapi root ga boleh diambil
Jika posisi sekarang belum diambil, assign ke kiri dan kanan mau ambil berapa dan root bebas mau ambil atau tidak

DP on Tree - Example

Apakah solusinya sudah selesai?



DP on Tree - Independent Knapsack

Kalian sudah bisa Independent Knapsack :D

Sekarang siap-siap buat soal berikutnya...

DP on Tree - Independent Knapsack II

Diberikan sebuah ~~binary~~ tree dengan N node. Tiap node bernilai V_i
Mau ambil paling banyak K node dimana setiap node yang dipilih merupakan independent set. Nilai dari K node yang dipilih adalah jumlah dari setiap nilai pada node tersebut. Berapa nilai maksimum yang dapat diperoleh?

$1 <= K <= N <= 100$
$1 <= V_i <= 10000$

DP on Tree - Independent Knapsack II

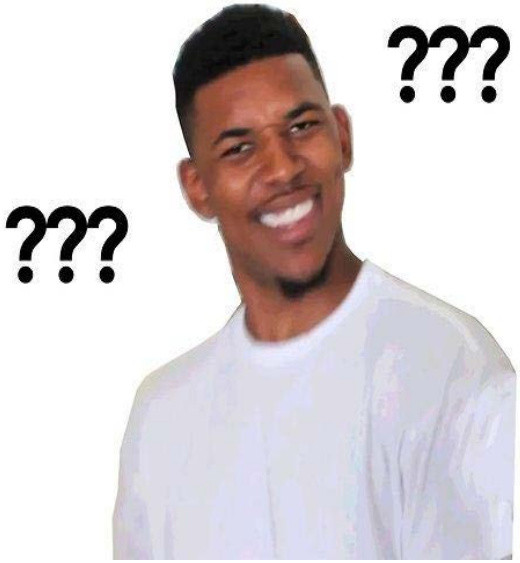
State : [now][status][K]
K disini sebagai penanda di subtree ini harus ambil K node

Base case
Di leaf, jika K beda status artinya gagal...

Transisi :
Jika posisi sekarang diambil, assign ke ~~kiri dan kanan~~ tiap anak mau ambil berapa tapi root ga boleh diambil
Jika posisi sekarang belum diambil, assign ke ~~kiri dan kanan~~ tiap anak mau ambil berapa dan root bebas mau ambil atau tidak

DP on Tree - Independent Knapsack II

Apakah solusinya sudah selesai?



DP on Tree - Independent Knapsack II

Transisi :

Jika posisi sekarang diambil, assign ke ~~kiri dan kanan~~ tiap anak mau ambil berapa tapi root ga boleh diambil

Jika posisi sekarang belum diambil, assign ke ~~kiri dan kanan~~ tiap anak mau ambil berapa dan root bebas mau ambil atau tidak

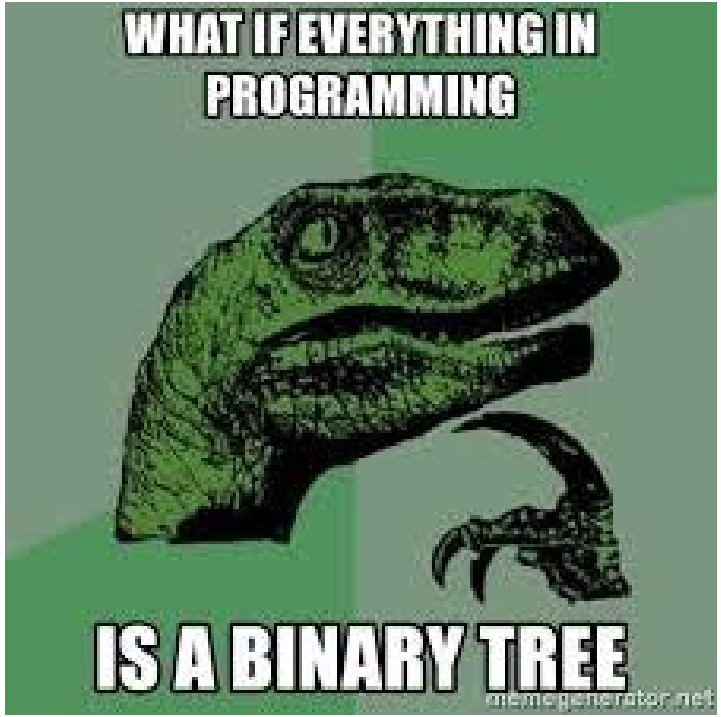
DP on Tree - Independent Knapsack II

Transisi :
Jika posisi sekarang diambil, assign ke ~~kiri dan kanan~~ tiap anak mau ambil berapa tapi root ga boleh diambil
Jika posisi sekarang belum diambil, assign ke ~~kiri dan kanan~~ tiap anak mau ambil berapa dan root bebas mau ambil atau tidak

Assign ke tiap anak...
Banyaknya cara = C(k+anak- 1,anak- 1) cara
EXPONENTIAL GUYS!!

DP on Tree - Independent Knapsack II

A meme dinosaur say….



DP on Tree - LCRS

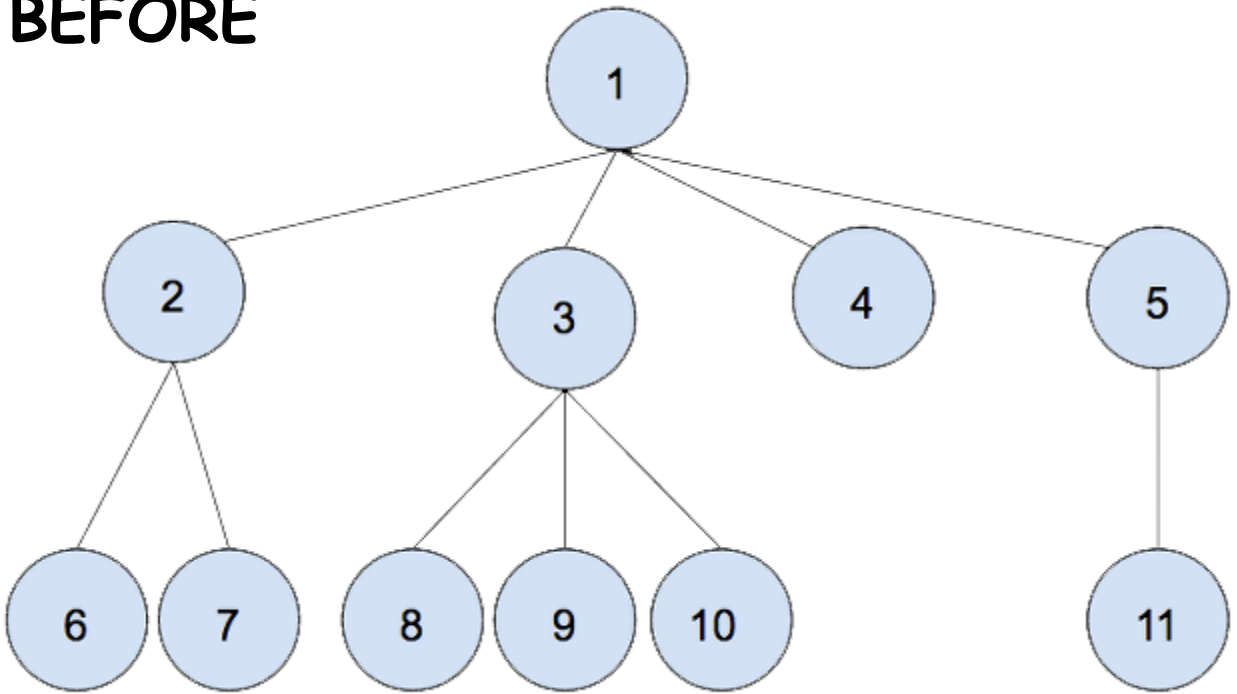
Welcome to another tree…

Left-Child Right-Sibling

Sama seperti namanya, anak kirinya adalah anak dari dia pada tree aslinya dan anak kanannya adalah sibling dari dia pada tree aslinya....

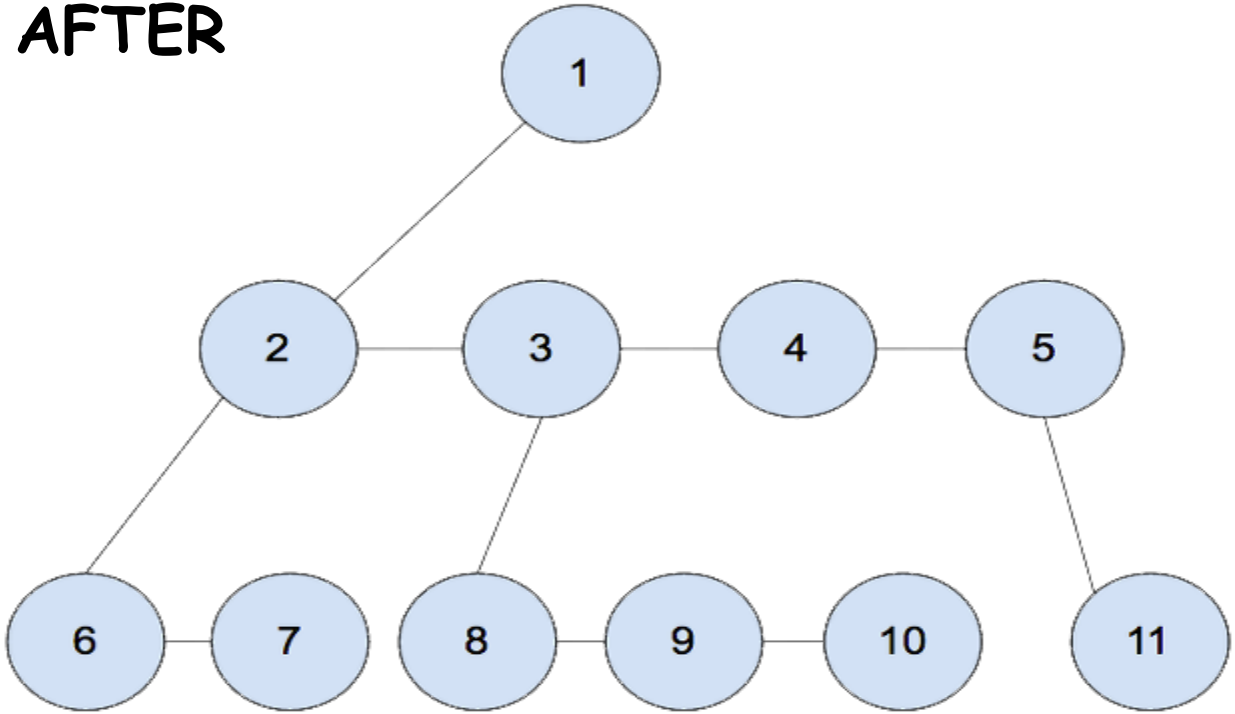
DP on Tree - LCRS

BEFORE



DP on Tree - LCRS

AFTER



DP on Tree - LCRS

Sudah jadi binary tree, sudah lebih mudah untuk di solve :D

DP on Tree - LCRS

Sudah jadi binary tree, sudah lebih mudah untuk di solve :D

Jangan lupa keep parentnya tiap node...

DP on Tree - LCRS

Gimana cara bikin LCRS nya??

DP on Tree - LCRS

Gimana cara bikin LCRS nya??

Simply mainin dari setiap child yang ada...

DP on Tree - Bottom Up

- Isi base case
- DFS child-childnya
- Child beres, gabungin sama dp sekarang (dp -> dp')
- dp <- dp'

DP on Tree - Bottom Up

```
// dp[now][k][status]
//set jadi -INF karena maks
set_min(dp[now]);
//base case
dp[now][0][0] = 0;
dp[now][1][1] = c[now];
for(int nex : child[now])
{
   dfs(nex);
   //copy dp[now] to temporary array (dp')
   copy(dp[now],temp);
   //reset dp[now]
   set_min(dp[now]);

   //untuk status 0
   for(int i=0;i<=k;i++)
      for(int j=0;j<=i;j++)
         dp[now][i][0] = max(dp[now][i][0] ,
                             temp[j][0] + max(dp[nex][i-j][0] , dp[nex][i-j][1]));
   //untuk status 1
   for(int i=0;i<=k;i++)
      for(int j=0;j<=i;j++)
         dp[now][i][1] = max(dp[now][i][1] ,
                             temp[j][1] + dp[nex][i-j][0]);
   //with above code, dp udah jadi dp'
}
```

Q & A


thank yous