Divide and Conquer

Jonathan Irvin Gunawan
Google Asia Pacific, Pte. Ltd.

Z

X C

V B N M

A S D F G H J K

Q W E R T Y U I O P V B X F R T L

prerequisite

tau rekursif

bawa kertas

(serius, ada beberapa latihan yang harus ngotret hari ini)

bisa ngitung

udah cuci muka tadi

pagi

oh sebelum mulai DnC, mau ngecek dulu apakah kalian udah tahu ini (karena lumayan related)

binary search

binary search the answer

kalau ada yang enggak, gw coba bahas di papan tanpa slide

definisi DnC

In computer science, **divide and conquer** (**D&C**) is an algorithm design paradigm based on multi-branched recursion. A divide and conquer algorithm works by recursively breaking down a problem into two or more sub-problems of the same or related type, until these become simple enough to be solved directly. The solutions to the sub-problems are then combined to give a solution to the original problem.

motivation problem

buat hari ini

dikasih N titik di bidang 2 dimensi, cari dua pasang titik yang jarak euclideannya paling deket

contoh paling klasik (dan paling textbook example)
DnC : sorting

merge sort

```
void mergesort(v) {
  if (v.size() == 1) return;  // base case

  // divide
  vector<int> a, b;
  a.insert(a.end(), v.begin(), v.begin() + v.size() / 2);
  b.insert(b.end(), v.begin() + v.size() / 2, v.end());
  mergesort(a);
  mergesort(b);

  // conquer
  int l = 0;
  int r = 0;
  for (int i = 0; i < v.size(); ++i) {
    if (l == a.size()) v[i] = b[r++];
    else if (r == b.size()) v[i] = a[l++];
    else if (a[l] < b[r]) v[i] = a[l++];
    else v[i] = b[r++];
  }
}
```

```
void mergesort(v) {
  if (v.size() == 1) return;  // base case

  // divide
  vector<int> a, b;
  a.insert(a.end(), v.begin(), v.begin() + v.size() / 2);
  b.insert(b.end(), v.begin() + v.size() / 2, v.end());
  mergesort(a);
  mergesort(b);

  // conquer
  v.clear();
  merge(a.begin(), a.end(), b.begin(), b.end(), v.begin());
}
```

mari kita coba analisa

kompleksitasnya

katakan f(N) adalah waktu yang dibutuhkan untuk ngemergesort N elemen. base casenya f(1) = 0

untuk N>1 ?

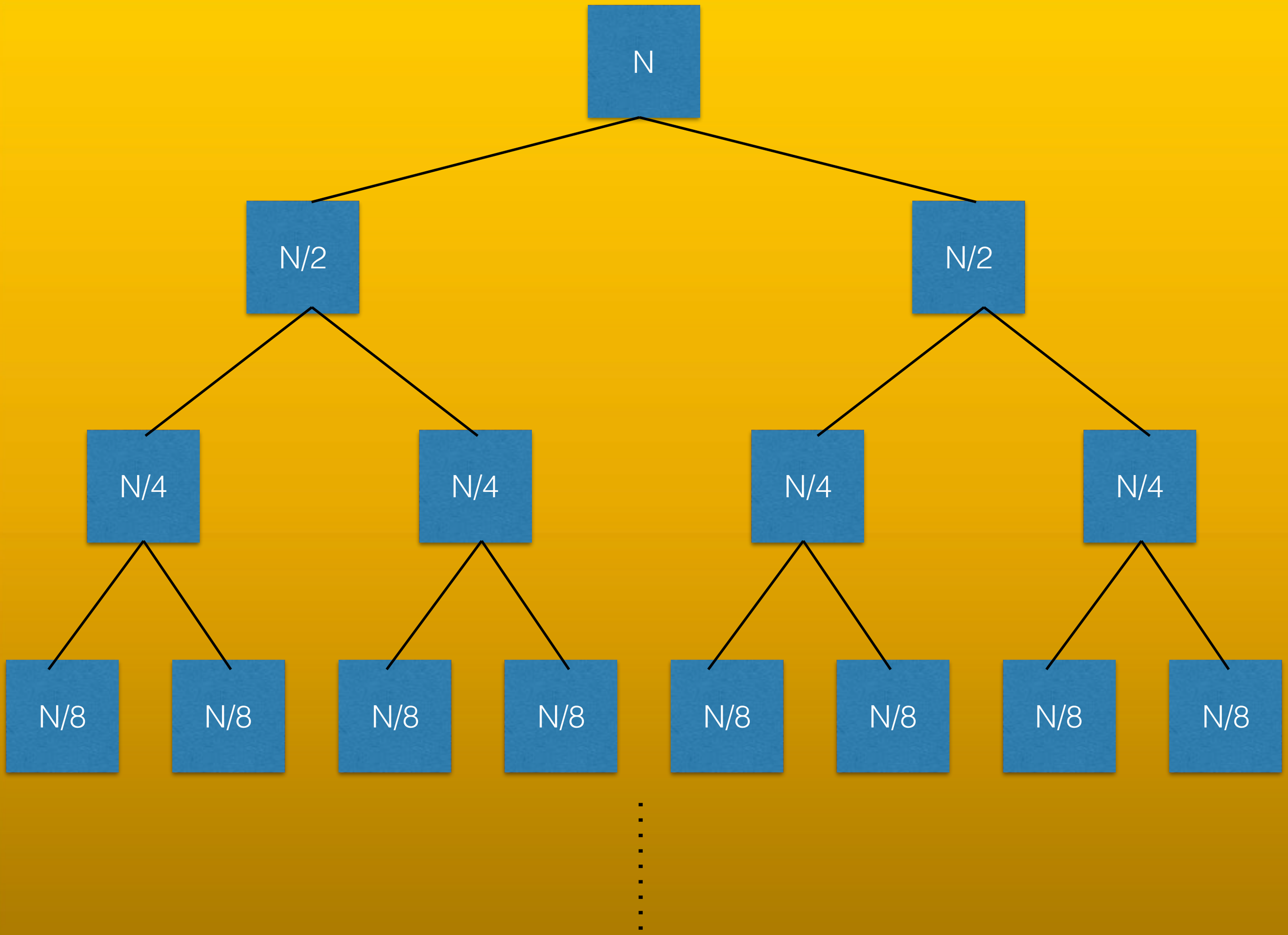
```
void mergesort(v) {
  if (v.size() == 1) return;  // base case

  // divide
  vector<int> a, b;
  a.insert(a.end(), v.begin(), v.begin() + v.size() / 2);  // N
  b.insert(b.end(), v.begin() + v.size() / 2, v.end());  // N
  mergesort(a);  // f(N/2)
  mergesort(b);  // f(N/2)

  // conquer  // N
  int l = 0;
  int r = 0;
  for (int i = 0; i < v.size(); ++i) {
    if (l == a.size()) v[i] = b[r++];
    else if (r == b.size()) v[i] = a[l++];
    else if (a[l] < b[r]) v[i] = a[l++];
    else v[i] = b[r++];
  }
}
```

so f(N) = 2f(N/2) + 3N

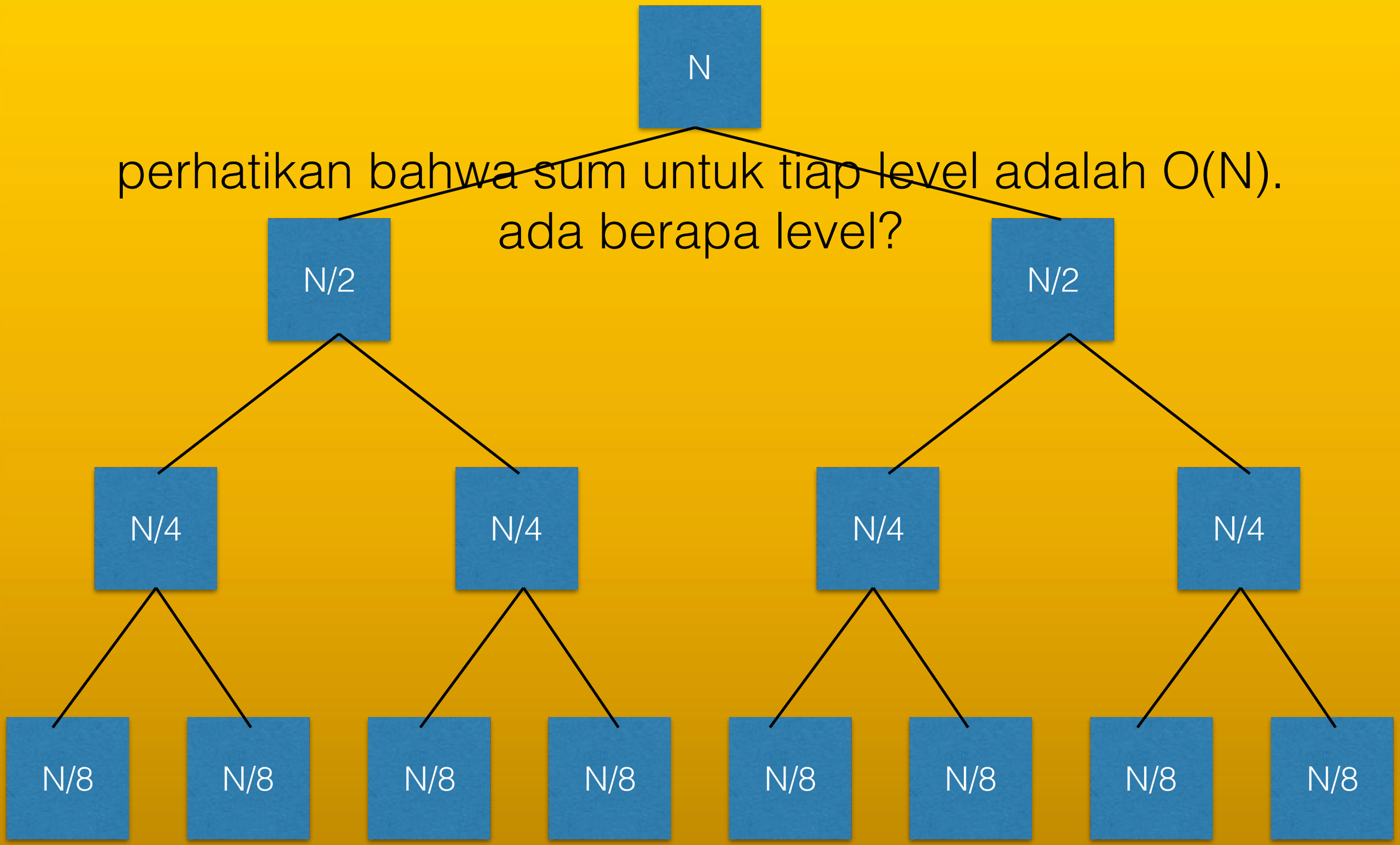
ato bisa disimplify jadi

f(N) = 2f(N/2) + O(N)

= .....


N
N/2
N/2
N/4
N/4
N/4
N/4
N/8
N/8
N/8
N/8
N/8
N/8
N/8
N/8

perhatikan bahwa sum untuk tiap level adalah O(N).
ada berapa level?



O(N) per level

ada O(lg N) level

jadinya f(N) = O(N lg N)

coba, latihan

kalau misalkan mergesort nya bagi jadi 3 vector a, b, c. mergesort masing2 vector, terus merge. berapa total kompleksitasnya?

$$f(N) = 3f(N/3) + O(N)$$

O(N) per level

ada O(lg_3 N) level

sama aja tetep f(N) = O(N lg N)

coba, latihan lagi langsung yang banyak
karena capek bikin slidenya, pembahasan tiap soal
gak ditulis slide, pake papan tulis aja

$$f(N) = 2f(N/2) + O(1)$$
$$f(N) = ?$$

$$f(N) = 2f(N/2) + O(N)$$
$$f(N) = ?$$

$$f(N) = 2f(N/2) + O(N^2)$$
$$f(N) = ?$$

coba, latihan lagi langsung yang banyak
karena capek bikin slidenya, pembahasan tiap soal
gak ditulis slide, pake papan tulis aja

$$f(N) = 2f(N/2) + O(1)$$
$$f(N) = O(N)$$

$$f(N) = 2f(N/2) + O(N)$$
$$f(N) = ?$$

$$f(N) = 2f(N/2) + O(N^2)$$
$$f(N) = ?$$

coba, latihan lagi langsung yang banyak
karena capek bikin slidenya, pembahasan tiap soal
gak ditulis slide, pake papan tulis aja

$$f(N) = 2f(N/2) + O(1)$$
$$f(N) = O(N)$$

$$f(N) = 2f(N/2) + O(N)$$
$$f(N) = O(N \lg N)$$

$$f(N) = 2f(N/2) + O(N^2)$$
$$f(N) = ?$$

coba, latihan lagi langsung yang banyak
karena capek bikin slidenya, pembahasan tiap soal
gak ditulis slide, pake papan tulis aja

$$f(N) = 2f(N/2) + O(1)$$
$$f(N) = O(N)$$

$$f(N) = 2f(N/2) + O(N)$$
$$f(N) = O(N \lg N)$$

$$f(N) = 2f(N/2) + O(N^2)$$
$$f(N) = O(N^2)$$

LAGI !!!

$$f(N) = 4f(N/2) + O(1)$$
$$f(N) = ?$$

$$f(N) = 4f(N/2) + O(N)$$
$$f(N) = ?$$

$$f(N) = 4f(N/2) + O(N^2)$$
$$f(N) = ?$$

LAGI !!!

$$f(N) = 4f(N/2) + O(1)$$
$$f(N) = O(N^2)$$

$$f(N) = 4f(N/2) + O(N)$$
$$f(N) = ?$$

$$f(N) = 4f(N/2) + O(N^2)$$
$$f(N) = ?$$

LAGI !!!

$$f(N) = 4f(N/2) + O(1)$$
$$f(N) = O(N^2)$$

$$f(N) = 4f(N/2) + O(N)$$
$$f(N) = O(N^2)$$

$$f(N) = 4f(N/2) + O(N^2)$$
$$f(N) = ?$$

LAGI !!!

$$f(N) = 4f(N/2) + O(1)$$
$$f(N) = O(N^2)$$

$$f(N) = 4f(N/2) + O(N)$$
$$f(N) = O(N^2)$$

$$f(N) = 4f(N/2) + O(N^2)$$
$$f(N) = O(N^2 \lg N)$$

TERUS SAMPE BOSEN !!!

$$f(N) = f(N/2) + O(1)$$
$$f(N) = ?$$

$$f(N) = f(N/2) + O(N)$$
$$f(N) = ?$$

$$f(N) = f(N/2) + O(N^2)$$
$$f(N) = ?$$

TERUS SAMPE BOSEN !!!

$$f(N) = f(N/2) + O(1)$$
$$f(N) = O(\lg N)$$

$$f(N) = f(N/2) + O(N)$$
$$f(N) = ?$$

$$f(N) = f(N/2) + O(N^2)$$
$$f(N) = ?$$

TERUS SAMPE BOSEN !!!

$$f(N) = f(N/2) + O(1)$$
$$f(N) = O(\lg N)$$

$$f(N) = f(N/2) + O(N)$$
$$f(N) = O(N)$$

$$f(N) = f(N/2) + O(N^2)$$
$$f(N) = ?$$

TERUS SAMPE BOSEN !!!

$$f(N) = f(N/2) + O(1)$$
$$f(N) = O(\lg N)$$

$$f(N) = f(N/2) + O(N)$$
$$f(N) = O(N)$$

$$f(N) = f(N/2) + O(N^2)$$
$$f(N) = O(N^2)$$

ini sebenernya ada rumusnya yang bisa either lu coba derive sendiri, ato ngapal

https://en.wikipedia.org/wiki/Master_theorem

ntar baca sendiri aja, intuisinya sama kayak yang udah dibahas.

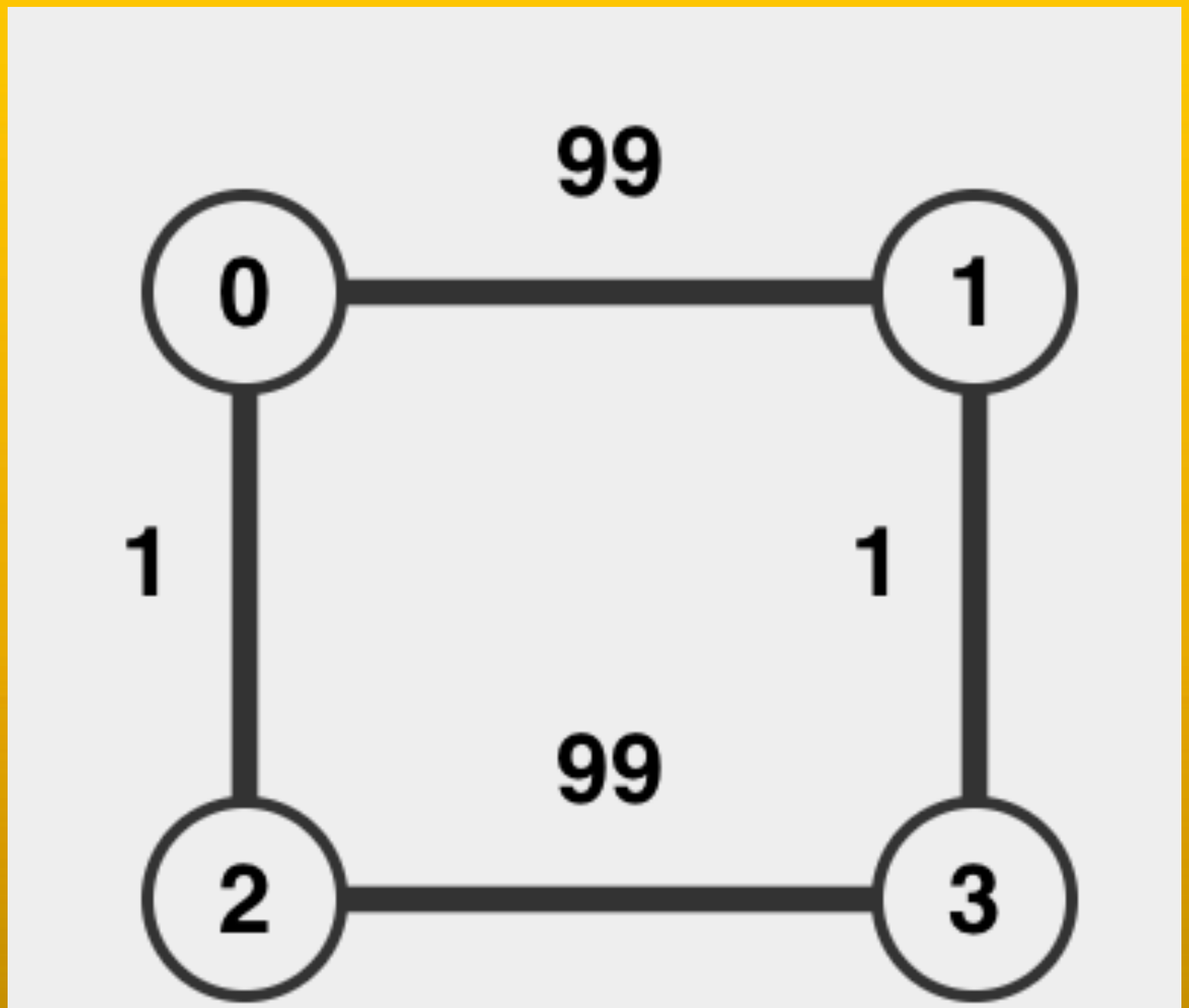
minimum spanning tree

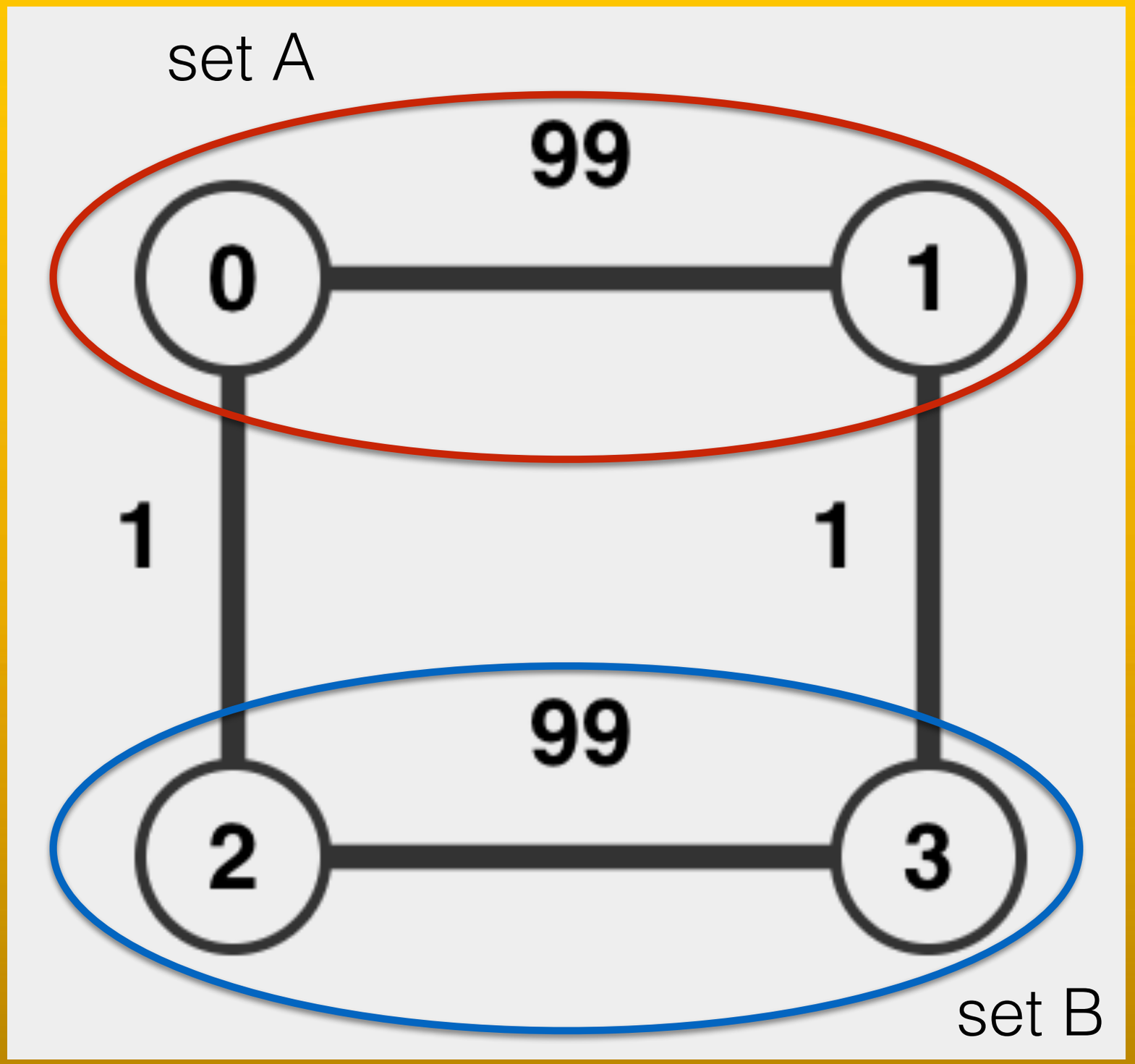
algo dnc :

bagi jadi dua set of node A dan B yang ukurannya hampir sama. cari mst dari masing-masing set, lalu hubungkan mereka dengan edge yang weightnya paling minimum yang menghubungkan A dan B

$$O((E+V) \lg V)$$

lalu kenapa kalian gak pernah denger algo ini?


99
0
1
1
1
99
2
3


set A
99
0
1
1
1
99
2
3
set B

salah total,

jadi ati2 juga kalo mau dnc

biasanya ngeproofnya bisa pake induksi

mari kita balik ke motivasi soal awal (closest pair problem)

jadi algonya, kalo lu diberikan N titik, lu bagi jadi dua set of titik A dan B, yang mana semua titik di A itu disebelah kirinya (ato sama dengan) semua titik di B.

lu cari closest pair di A sama di B, lalu ambil yang lebih kecil

```
double closest_pair(S) {
  vector<points> A, B;
  A.insert(A.end(), S.begin(), S.begin() + S.size() / 2);
  B.insert(B.end(), S.begin() + S.size() / 2, S.end());
  return min(closest_pair(A), closest_pair(B));
}
```

jadi

$$f(N) = 2f(N/2) + O(1)$$

jadi

$$f(N) = O(N)$$

ngaco, tapi bisa aja kan closest pair nya ternyata (a, b) yang mana a itu di dalam A dan b itu di dalam B

jadi, lu juga harus cek untuk semua pasang node (a, b) yang mana a di dalam A dan b di dalam B

```
double closest_pair(S) {
  vector<points> A, B;
  A.insert(A.end(), S.begin(), S.begin() + S.size() / 2);
  B.insert(B.end(), S.begin() + S.size() / 2, S.end());
  double ret = min(closest_pair(A), closest_pair(B));
  for (points a : A) {
    for (points b : B) {
      ret = min(ret, dist(a, b));
    }
  }
  return ret;
}
```

jadi

$$f(N) = 2f(N/2) + O((N/2)^2)$$

$$f(N) = 2f(N/2) + O(N^2)$$

jadi

$$f(N) = O(N^2)$$

sama aja dong kayak

algoritma naif biasa :”””((((

optimisasi dewa : untuk setiap node a di A, lu cuma cek node b di B yang jaraknya ≤ min(closest_pair(A), closest_pair(B)) dari a.

dengan optimisasi ini, setiap titik di A paling banyak cuma perlu cek 6 titik doang

proofnya kasar aja pake gambar di papan tulis, males bikin gambar di slide

detil implementasinya, lu harus ngesort2 titik2nya gitu supaya lu gak perlu loop mahal2 buat cari 6 titiknya

detil implementasinya left as your own exercise karena gak masuk dalam fokus analisis dncnya :””””

sekarang jadi

$$f(N) = 2f(N/2) + 6N$$

$$= 2f(N/2) + O(N)$$

jadi

$$f(N) = O(N \lg N)$$

summary kalo mau dnc

1. tentuin base case (biasanya gak susah)
2. tentuin rekursif sub-problemnya gimana
3. tentuin gabunginnya gimana
4. coba dapetin intuisi correctness (pake induksi biasanya)
5. analisis running time pake recursion tree, ato master theorem

meet in the middle

inti ide dari meet in the middle :

lu complete search dari dua “arah” atau dua “sumber”, maka lu akan dapat dua huge list (huge karena complete search)

lalu lu cek apakah dua list ini “intersect” in some way

contoh, misalkan lu punya graph (super besar) yang dijamin tiap node degreenya gak lebih dari 15,

lu ditanya apakah shortest path dari S ke T gak lebih dari 8.

kalo lu BFS dari S sebanyak 8 step, lu akan visit 15^8=2.562.890.625 nodes

jadi, caranya lu BFS dari S dan T masing2 4 steps, masing2 lu akan visit 15^4 = 50.625 nodes

kalo dari dua set of nodes ini ada intersection, maka S ke T bisa dalam 8 steps

contoh lain

subset sum

diberikan set N angka, tentukan apakah ada subset yang sumnya tepat K.

$1 \leq N \leq 36$

$1 \leq K \leq 1.000.000.000$

solusi : bagi set jadi dua subset, A dan B, masing2 sizenya N/2.

untuk semua subset A dan B, hitung jumlahnya.

lu akan punya list of sum dari A sebanyak 2^(N/2) dan list of sum dari B sebanyak 2^(N/2).

lu cek apakah ada pair (a, b) yang mana a dari A dan b dari B, yang memenuhi a + b = K

bisa pake sliding window supaya linear terhadap max(size(A), size(B))

jadi kompleksitasnya
O(2^(N/2) lg (2^(N/2)) <— buat ngesort)
= O(2^(N/2)*N)

2^(36/2) * 36 = 9.437.184

POSSIBBRUUUUUUU

tips soal meet in the middle

kalo konstrainnya “nanggung”, gak bisa di 2^N tapi masih terlalu kecil buat polynomial solution (< 50), coba mikir2 solusi 2^(N/2)

nah sekarang latihan soal

diberikan N interval (L, R). ceritanya untuk setiap range (L, R), lu masukin semua integer x yang memenuhi $L \leq x \leq R$ ke array A.

berapa elemen terkecil ke-k di array A

$1 \leq N \leq 100.000$

$1 \leq L \leq R \leq 2.000.000.000$

$1 \leq k \leq$ banyaknya elemen di A

binary search the answer

untuk setiap bilangan x, lu bisa hitung ada berapa bilangan di A yang ≤ x dalam O(N). katakan itu adalah f(x).

lu binary search the answer buat cari x terkecil yang memenuhi f(x) ≥ k.

total jadi O(N lg (2.000.000.000))

latihan soal lagi

ada 3 array A, B, C, ukurannya N

ada tiga variabel juga, totA, totB, dan totC

untuk setiap $1 \leq i \leq N$, lu boleh pilih antara tiga ini :
1. totA += A[i], totB += B[i]
2. totB += B[i], totC += C[i]
3. totA += A[i], totC += C[i]

tentukan apakah mungkin lu milihnya sedemikian sehingga totA = totB = totC

$1 \leq N \leq 25$

meet in the middle

generate semua kemungkinan 3^13 buat 13 half atas, dan generate semua kemungkinan 3^12 buat 12 half bawah

cek apakah ada dua kemungkinan di masing2 half yang kalo “digabungin” bakal memenuhi syarat

3^13 = 1.594.323 oke sip can

oke terakhir, ini agak susah

juri punya X[0], X[1], … X[N-1]. X sorted menaik. lu mau cari gap yang lebih panjang ato sama dengan gap rata2

dengan kata lain, let g[i] = x[i+1] - x[i] untuk $0 \le i < N-1$

lu mau cari j yang memenuhi $g[j] \ge (sum(g) / (N - 1))$

$$1 \leq N \leq 1e18$$

binary search, ambil M = (L + R) / 2.

cari average kiri sama average kanan, recurse ke yang lebih besar

cara cari average y[a..b-1] bisa tinggal (x[b] - x[a]) / (b - a)

O(lg N)

EOF

Q&A?