1. (BOBOT: 5 + 10)
2. Prinsip dari Algoritma Divide & Conquer

Algoritma divide and conquer sesuai dengan namanya yaitu membagi (divide) masalah menjadi lebih kecil, ukuran pembagiannya harus mirip sama masalah sebelumnya (idealnya ukurannya sama) atau biasa disebut dengan tahapan dekomposisi. Kemudian conquer yaitu menyelesaikan sub-sub masalah. Terakhir, dilakukan combine gabungkan solusi-solusi kecil jadi solusi permasalahan semula.

Algoritma ini sifatnya rekursif jadi secara kompleksitas tentu akan tinggi (Big O nya).

Banyak persoalan yang dapat diselesaikan oleh algoritma ini, diantaranya:

* Pangkat
* Sorting (Merge/Quick)
* Closest Pair
* Perkalian Matriks

1. Hasil pengurutan di bawah ini menggunakan algoritma Divide & Conquer 🡪 Merge Sort

* Tahap Divide
* Tahap Conquer
* Tahap Combine

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  | 46 | 37 | 86 | 23 | 52 | 22 | 65 | 74 | 25 |  |  |  |  |  |  |  |  | Masalah | |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  | 46 | 37 | 86 | 23 | 52 |  | 22 | 65 | 74 | 25 |  |  |  |  |  |  |  | Divide | |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  | 46 | 37 | 86 |  | 23 | 52 |  | 22 | 65 |  | 74 | 25 |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | 46 | 37 |  | 86 |  | 23 |  | 52 |  | 22 |  | 65 |  | 74 |  | 25 |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 46 |  | 37 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | 46 | 37 |  | 86 |  | 23 |  | 52 |  | 22 |  | 65 |  | 74 |  | 25 |  |  |  | Conquer | |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  | 86 | 46 | 37 |  |  | 52 | 23 |  |  | 65 | 22 |  | 74 | 25 |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  | 86 | 52 | 46 | 37 | 23 |  |  | 74 | 65 | 25 | 22 |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  | 86 | 74 | 65 | 52 | 46 | 37 | 25 | 23 | 22 |  |  |  |  |  |  |  |  | Combine | |

1. (BOBOT: 5 + 20)
2. Cara kerja dari Algoritma Branch & Bound

* Menggunakan struktur tree
* Pencarian BFS (namun bukan FIFO)
* Biasanya digunakan untuk perosalan optimasi, pencarian nilai maksimal atau minimal. Tentunya ada constraint yaitu jika tidak ada jalan maka tidak dilajutkan.

1. Tahapan B&B

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| AB tidak ada jalur | | | | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | A | B | C | D |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| A | - | - | 10 | 8 |  | - | - | 10 | 8 |  | - | - | 2 | 0 |  | - | - | 2 | 0 |
| B | 11 | - | 7 | 10 |  | 11 | - | 7 | 10 |  | 11 | - | 7 | 10 |  | 4 | - | 0 | 3 |
| C | 13 | 9 | - | 14 |  | 13 | 9 | - | 14 |  | 13 | 9 | - | 14 |  | 13 | 9 | - | 14 |
| D | 8 | 11 | 12 | - |  | 8 | 11 | 12 | - |  | 8 | 11 | 12 | - |  | 8 | 11 | 12 | - |
|  |  |  |  |  |  |  |  |  |  |  |  | R= | 8 |  |  |  | R= | 8+7 |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  | - | - | 2 | 0 |  | - | - | 2 | 0 |  |  |  |  |  |
|  |  |  |  |  |  | 4 | - | 0 | 3 |  | 4 | - | 0 | 3 |  |  |  |  |  |
|  |  |  |  |  |  | 4 | 0 | - | 5 |  | 4 | 0 | - | 5 |  |  |  |  |  |
|  |  |  |  |  |  | 8 | 11 | 12 | - |  | 0 | 3 | 4 | - |  |  |  |  |  |
|  |  |  |  |  |  |  | R= | 8+7+9 |  |  |  | R= | 8+7+9+8 | |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  | A | row | 1 |  |  |  |  |  |  |  |  |  |  |  |  |
| RCM node 1 | | |  |  | B | col | 2 |  |  |  |  |  |  |  |  |  |  |  |  |
| - | - | 2 | 0 |  | - | - | - | - |  |  |  |  |  |  |  |  |  |  |  |
| 4 | - | 0 | 3 |  | - | - | 0 | 3 |  |  |  |  |  |  |  |  |  |  |  |
| 4 | 0 | - | 5 |  | 4 | - | - | 5 |  |  |  |  |  |  |  |  |  |  |  |
| 0 | 3 | 4 | - |  | 0 | - | 4 | - |  |  |  |  |  |  |  |  |  |  |  |
|  | b= | 32 | |  |  | b= | 35 | |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  | b= | 35 | +- | tidak ada jalur | | |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  | **A** | row | 1 |  |  | A | row | 1 |  |  |  |  |  |  |  |
| RCM node 1 | | |  |  | **C** | col | 3 |  |  | C | col | 3 |  |  |  |  |  |  |  |
| - | - | 2 | 0 |  | - | - | - | - |  | - | - | - | - |  |  |  |  |  |  |
| 4 | - | 0 | 3 |  | 4 | - | - | 3 |  | 4 | - | - | 0 |  |  |  |  |  |  |
| 4 | 0 | - | 5 |  | - | 0 | - | 5 |  | - | 0 | - | 5 |  |  |  |  |  |  |
| 0 | 3 | 4 | - |  | 0 | 3 | - | - |  | 0 | 3 | - | - |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  | b= | 35 | | +2 |  |  |  |  |  |
|  | b= | 32 | |  |  | b= | 35 | |  |  | b= | 37 | |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  | A | row | 1 |  |  | A | row | 1 |  |  |  |  |  |  |  |
| RCM node 1 | | |  |  | D | col | 4 |  |  | D | col | 4 |  |  |  |  |  |  |  |
| - | - | 2 | 0 |  | - | - | - | - |  | - | - | - | - |  |  |  |  |  |  |
| 4 | - | 0 | 3 |  | 4 | - | 0 | - |  | 4 | - | 0 | - |  |  |  |  |  |  |
| 4 | 0 | - | 5 |  | 4 | 0 | - | - |  | 4 | 0 | - | - |  |  |  |  |  |  |
| 0 | 3 | 4 | - |  | - | 3 | 4 | - |  | - | 0 | 4 | - |  |  |  |  |  |  |
|  | b= | 32 | |  |  | b= | 35 | |  |  | b= | 39 | |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  | A | row | 1 |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  | D | col | 4 |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  | - | - | - | - |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  | 0 | - | 0 | - |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  | 4 | 0 | - | - |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  | - | 0 | 4 | - |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  | b= | 39 | |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  | b= | 39 | +0 |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  | b= | 39 |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  | **C** | row | 3 |  |  |  |  |  |  |  |  |  |  |  |  |
| RCM AC | |  |  |  | **B** | col | 2 |  |  |  |  |  |  |  |  |  |  |  |  |
| - | - | - | - |  | - | - | - | - |  |  |  |  |  |  |  |  |  |  |  |
| 4 | - | - | 0 |  | 4 | - | - | 0 |  |  |  |  |  |  |  |  |  |  |  |
| - | 0 | - | 5 |  | - | - | - | 5 |  |  |  |  |  |  |  |  |  |  |  |
| 0 | 3 | - | - |  | 0 | - | - | - |  |  |  |  |  |  |  |  |  |  |  |
| b= | 37 | |  |  |  | b=37 | +0 |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  | b=37 |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  | C | row | 3 |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  | D | col | 4 |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  | b= | 37 | | +5 | sudah tidak mungkin minimum | | | | | |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Solusi TSP | |  |  |  |  |  | A | B | C | D |  |  |  |  |  |  |  |  |  |
| Lintasan yang dihasilkan | | | | |  | A | - | - | 10 | 8 |  |  |  |  |  |  |  |  |  |
| AC | 10 |  |  |  |  | B | 11 | - | 7 | 10 |  |  |  |  |  |  |  |  |  |
| CB | 9 |  |  |  |  | C | 13 | 9 | - | 14 |  |  |  |  |  |  |  |  |  |
| BD | 10 |  |  |  |  | D | 8 | 11 | 12 | - |  |  |  |  |  |  |  |  |  |
| DA | 8 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Sum | 37 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

Solusi

AC 🡪 CB 🡪 BD 🡪 DA dengan total lintasan 37

Gambar search tree

Shape, circle

Description automatically generated

1. (BOBOT : 15 + 5 + 15)
2. Solusi optimal dari Program Dinamis

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| w | p |  | W=6 |  |
| 2 | 10 |  |  |  |
| 3 | 12 |  |  |  |
| 3 | 9 |  |  |  |
| 2 | 14 |  |  |  |
|  |  |  |  |  |
| y | f0(y) | 10+f0(y-2) | f1(y) | xi |
| 0 | 0 | -inf | 0 | (0,0,0,0) |
| 1 | 0 | -inf | 0 | (0,0,0,0) |
| 2 | 0 | 10 | 10 | (1,0,0,0) |
| 3 | 0 | 10 | 10 | (1,0,0,0) |
| 4 | 0 | 10 | 10 | (1,0,0,0) |
| 5 | 0 | 10 | 10 | (1,0,0,0) |
| 6 | 0 | 10 | 10 | (1,0,0,0) |
|  |  |  |  |  |
| y | f1(y) | 12+f1(y-3) | f2(y) | xi |
| 0 | 0 | -inf | 0 | (0,0,0,0) |
| 1 | 0 | -inf | 0 | (0,0,0,0) |
| 2 | 10 | -inf | 10 | (1,0,0,0) |
| 3 | 10 | 12 | 12 | (0,1,0,0) |
| 4 | 10 | 12 | 12 | (0,1,0,0) |
| 5 | 10 | 22 | 22 | (1,1,0,0) |
| 6 | 10 | 22 | 22 | (1,1,0,0) |
|  |  |  |  |  |
| y | f2(y) | 9+f2(y-3) | f3(y) | xi |
| 0 | 0 | -inf | 0 | (0,0,0,0) |
| 1 | 0 | -inf | 0 | (0,0,0,0) |
| 2 | 10 | -inf | 10 | (1,0,0,0) |
| 3 | 12 | 9 | 12 | (0,1,0,0) |
| 4 | 12 | 9 | 12 | (0,1,0,0) |
| 5 | 22 | 19 | 22 | (1,1,0,0) |
| 6 | 22 | 21 | 22 | (1,1,0,0) |
|  |  |  |  |  |
| y | f3(y) | 14+f3(y-2) | f4(y) | xi |
| 0 | 0 | -inf | 0 | (0,0,0,0) |
| 1 | 0 | -inf | 0 | (0,0,0,0) |
| 2 | 10 | 14 | 14 | (0,0,0,1) |
| 3 | 12 | 14 | 14 | (0,0,0,1) |
| 4 | 12 | 24 | 24 | (1,0,0,1) |
| **5** | **22** | **26** | **26** | **(0,1,0,1)** |
| **6** | **22** | **26** | **26** | **(0,1,0,1)** |

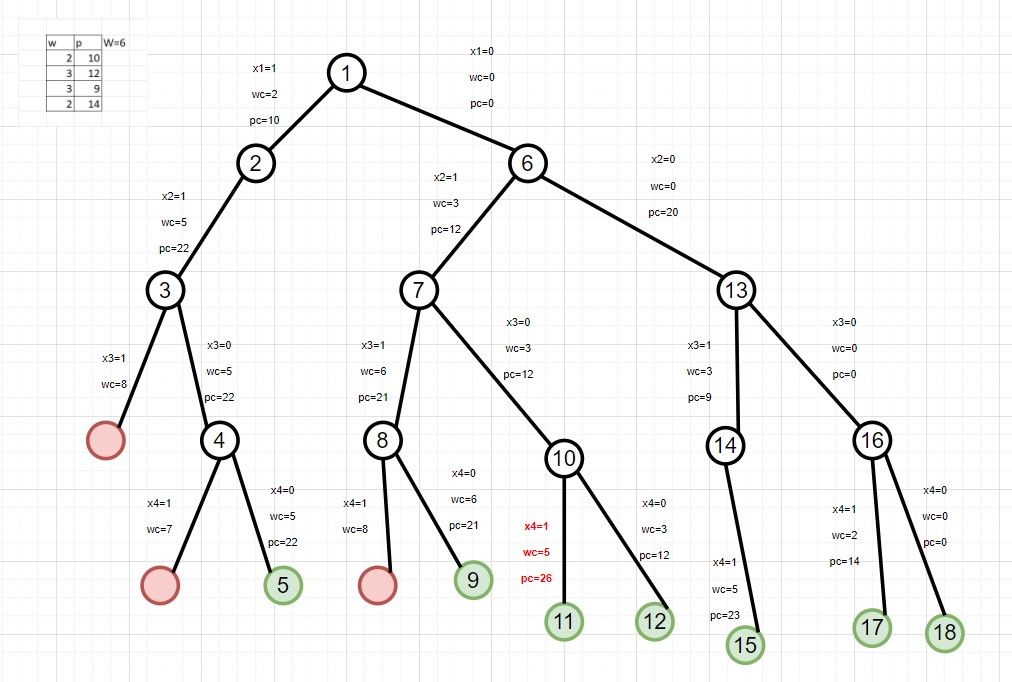
Solusi optimal dari program dinamis dengan profit 26 dan barang yang diambil (0,1,0,1)

1. Prinsip dari algoritma Backtracking

* Backtracking menggunakan Depth First Search atau biasa disebut pencarian mendalam untuk mencari solusi. Algoritma ini merupakan perbaikan dari algoritma brute force.
* Untuk menerapkan algoritma ini ke dalam knapsack 0/1 maka akan dihasilkan binary tree beserta generator functionnya yaitu Xn = 1 atau Xn = 0 di mana n merupakan barang yang akan diambil diberi nilai 1 dan tidak diambil diberi nilai 0.
* Kemudian kita juga perlu fungsi pembatas yaitu kapasitas, apakah fungsi ini masih True (kapasitas dari barang yang diambil tidak melebihi kapasitas maksimum) atau False. Ketika fungsi pembatas bernilai False maka simpul tersebut akan menjadi simpul mati, dan pencarian akan balik satu tahap satu tahap.
* Proses pencarian berhetin bila sudah tidak ada lagi jalan lain/sudah ada solusi (simpul hidup).
* Secara Big O nya sama seperti brute force yaitu O(p(n)2n) atau O(g(n)(n!) tapi keunggulannya adalah bisa memotong/pruning langkah dan mencoba kemungkinan lainnya

1. Solusi optimal dari Algoritma Backtracking

Pohon Pencariannya



\*) Untuk lingkaran merah: dead node, untuk lingkaran hijau: leaf node.

Solusi

Node 11 🡪 (0,1,0,1) dengan weight current = 5 dan profit current = 26.

1. (BOBOT : 5 + 5 + 15)
2. Kelebihan Algoritma Boyer Moore untuk string matching dibandingkan brute force

Kekurangan Algoritma Boyer Moore untuk string matching dibandingkan brute force

* Boyer More lebih cepat untuk alfabet besar (bisa character jump)
* Worst case : O(nm+A) 🡪 ditambah ukuran alfabet misalkan untuk searching English Text.
* Lambat untuk alfabet kecil (biner)

1. Hasil dari fungsi Lo

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| P | a | b | a | c | a |  | else=-1 | |
| LO | 0 | 1 | 2 | 3 | 4 |  |  |  |

1. Tahapan proses pencarian P = abaca dan T = cabdaecdabcadcabacada

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
|  | c | a | b | d | a | e | c | d | a | b | c | a | d | c | a | b | a | c | a | d | a |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | a | b | a | c | a |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  | lo=-1 | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  | i=8 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  | j=4 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  | a | b | a | c | a |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  | lo=-1 | |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  | i=12 |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  | j=4 |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  | a | b | a | c | a |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  | lo=-1 | |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  | i=17 |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  | j=4 |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  | a | b | a | c | a |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | lo=3 |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | i=18 |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | j=4 |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | a | b | a | c | a |  |  |