IMPLEMENTASI KNAPSACK PROBLEM MENGGUNAKAN ALGORITMA BREADTH FIRST SEARCH PADA OPTIMASI PEMILIHAN BUAH

Disusun untuk Memenuhi Tugas Akhir Mata Kuliah Kecerdasan Buatan

Dibimbing oleh Ibu Dr. Eng. Anik Nur Handayani, S.T., M.T.

Oleh:

Saddam Sinatrya Jalu Mukti 190535632609

UNIVERSITAS NEGERI MALANG

FAKULTAS TEKNIK

JURUSAN TEKNIK ELEKTRO

PRODI S1 TEKNIK INFORMATIKA

DESEMBER 2021

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I
PENDAHULUAN

1.1 Latar Belakang

Dalam membuat parsel buah, pengrajin akan mengumpulkan buah terbaik yang memiliki kualitas dan harga jual dan tinggi ke dalam satu parsel. Sehingga harga parsel buah akan memiliki nilai jual yang tinggi. Masalah tersebut akan timbul dalam pemilihan beberapa obyek yang harus ditampung ke dalam keranjang. Pada dasarnya keranjang memiliki batas maksimal, sehingga tidak semua buah dapat ditampung. Selain itu, perlu adanya pengaturan komposisi buah, sehingga fungsi keranjang dapat digunakan secara optimal dan mendapatkan keuntungan dari harga jual secara maksimal. Dari permasalahan tersebut, muncullah suatu permasalahan yang disebut *knapsack problem* atau permasalahan ransel.

Secara sederhana *knapsack problem* termasuk ke dalam optimasi kombinasi. Optimasi dapat dicapai dengan berbagai metode misalnya: *brute-force*, *backtracking*, *greedy*, *dynamic problem*, dan algoritma genetika. Metode yang digunakan pada makalah ini adalah metode *brute-force* dengan penjelajahan graf BFS (*Breadth-First Search*). Metode ini termasuk ke dalam *blind search* atau *uninform search*. Sehingga memiliki performa yang lambat, namun cukup baik pada skala kecil.

1.2 Rumusan Masalah

Rumusan masalah pada makalah ini adalah bagaimana cara menentukan komposisi berbagai jenis buah-buahan ke dalam keranjang parsel secara tepat tanpa melebihi batas maksimum kapasitas keranjang untuk mendapatkan keuntungan maksimal.

1.3 Manfaat

Manfaat makalah bagi penulis adalah untuk memperluas wawasan dari keilmuan yang telah dipelajari dengan mengkaji permasalahan pada implementasi metode *brute-force* BFS pada permasalahan *knapsack*. Manfaat makalah bagi pembaca adalah sebagai jalur untuk memperluas wawasan dan pengetahuan tentang implementasi metode *brute-force* BFS pada permasalahan *knapsack*.

BAB II
TINJAUAN PUSTAKA

2.1 *Knapsack Problem*

Knapsack problem adalah sebuah permasalahan dimana seseorang dihadapkan pada permasalahan optimasi pada pemilihan benda yang dapat dimasukkan ke dalam wadah yang memiliki keterbatasan ruang atau daya tampung (Supriana, 2016). Contohnya jika terdapat berbagai macam barang yang harus dimasukkan ke dalam kontainer namun memiliki batas kapasitas maksimum sehingga tidak semua barang dapat dimasukkan. Oleh karena itu, perlu dipilih kombinasi barang yang memiliki keuntungan optimum namun tidak melebihi batas maksimum.

Knapsack problem atau permasalahan keranjang telah dipelajari lebih dari satu abad dengan karya diterbitkan pada tahun 1897. Namun David (Pisinger, 1995) menjelaskan bahwa istilah *knapsack problem* itu sendiri baru ditemukan oleh George Dantzig pada tahun 1957 dan digunakan untuk mengacu pada masalah biasa dalam mengemas barang-barang yang paling berharga atau berguna tanpa membebani keranjang.

Pada permasalahan ini terdapat dua variabel yaitu daftar barang dan kapasitas maksimal dari keranjang (W). Setiap item memiliki properti keuntungan pi dan berat wi. Permasalahan ini fokus pada pemilihan *subset* dari sejumlah n barang dengan keuntungan yang dimaksimalkan tanpa melebihi batas kapasitas maksimal. Masalah ini dapat diformulasikan sebagai berikut.

maksimalkan $\sum_{i=1}^{n} p_i x_i$

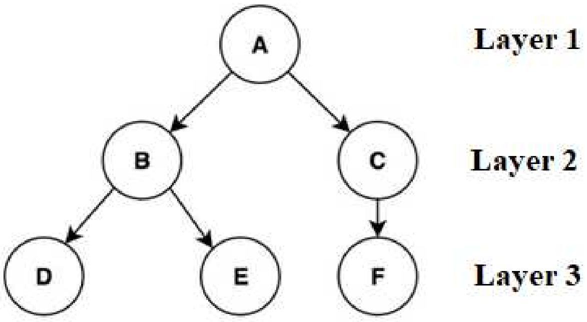
dengan syarat $\sum_{i=1}^{n} w_i x_i < W$ dan $x_i \in \{0,1\}$

Variabel x_i merupakan variabel biner yang apabila bernilai 1, maka barang akan dimasukkan ke dalam ransel. Sebaliknya apabila bernilai 0, maka barang tidak dimasukkan ke dalam ransel.

2.2 *Breadth-First Search*

Knapsack problem dapat diselesaikan dengan berbagai macam algoritma, salah satunya menggunakan algoritma *brute-force* berbasis *Breadth-First Search* atau BFS. Menurut Robbi (Rahim et al., 2018) *Breadth-First Search* adalah algoritma yang melakukan pencarian melebar dengan mengunjungi simpul sesuai urutan kedalaman. Metode BFS didasarkan pada pencarian dengan menjelajahi akarnya dari sebuah pohon hingga tidak memiliki akar. Algoritma ini menjadi landasan utama dari algoritma penjelajahan graf lain.

Dalam BFS, satu simpul dipilih untuk dikunjungi dan ditandai telah dikunjungi. Setiap cabang dari simpul dimasukkan ke dalam antrean. Kemudian tetangganya dikunjungi dan proses diulang kembali. Sehingga dapat dikatakan bahwa algoritma BFS bekerja dengan melakukan proses penjelajahan dari *layer* 1 ke $layer\ n$. Sebagai ilustrasi, pada gambar graf di bawah ini proses antrean yang dihasilkan adalah $A \rightarrow B \rightarrow C \rightarrow D \rightarrow E \rightarrow F$.



Gambar 2.1 Contoh Penjelajahan BFS

Arie (Lumenta, 2014) mengategorikan pencarian metode *Breadth-First Search* sebagai tanpa informasi atau dikenal sebagai *uninform search* dan *blind search*. Hal ini disebabkan karena untuk menerapkan metode BFS tidak membutuhkan informasi heuristik yang membantu proses penjelajahan. Berikut adalah langkah-langkah dalam pencarian dengan metode BFS.

```
procedure BFS(G, root):
  let Q be a queue
  label root as explored
  Q.enqueue(root)
  while Q is not empty do
    v := Q.dequeue()
    if v is the goal then
      return v
    for all edges from v to w in G.adjacentEdges(v) do
      if w is not labeled as explored then
        label w as explored
        Q.enqueue(w)
```

Implementasi dari algoritma ini sangat luas, misalnya algoritma ini dapat diimplementasikan pada permainan catur. Komputer dapat membangun pohon dari kemungkinan gerakan dan menggunakan metode pencarian BFS untuk menentukan gerakan terbaik. Namun semakin dalam pohon, maka semakin besar memori yang diperlukan. Sehingga mesin catur harus membatasi kedalaman pohon.

2.3 *State of the Art*

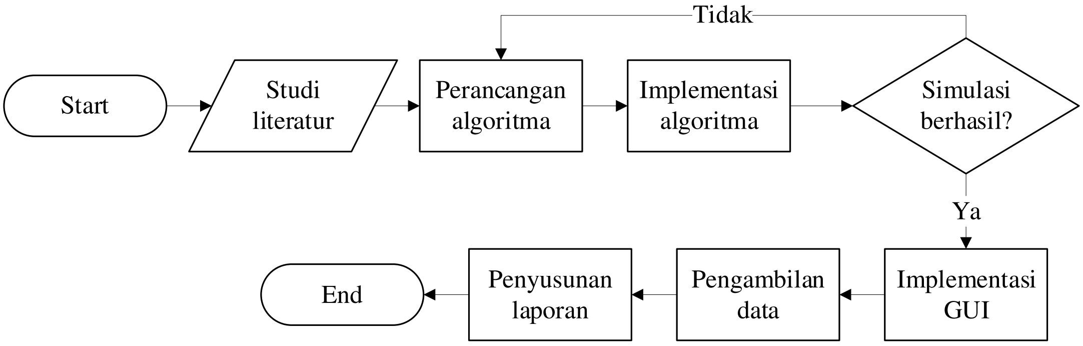
Penelitian ini digunakan untuk menyelesaikan *knapsack problem* yang pendekatan sebelumnya tidak bersifat *completeness* atau tidak dapat menjamin penemuan solusi terbaik dari himpunan solusi yang ada. Penelitian terdahulu digunakan untuk menganalisis dan memperluas pembahasan penelitian, serta membedakannya dengan penelitian yang telah dilakukan. Dalam makalah ini disertakan dua jurnal penelitian sebelumnya yang berhubungan dengan *knapsack problem*. Jurnal tersebut antara lain:

1. Penelitian dengan judul *Implementasi Algoritma Genetika Pada Knapsack Problem Untuk Optimasi Pemilihan Buah Kemasan Kotak*. Diambil dari *Seminar Nasional Aplikasi Teknologi Informasi 2010*, diteliti oleh Komang Setemen pada tahun 2010 di Yogyakarta. Penelitian ini membahas bagaimana *knapsack problem* dapat diselesaikan dengan algoritma genetika. Penggunaan algoritma ini dinilai cukup baik pada *knapsack problem*. Namun hasil program bergantung pada parameter genetika, contohnya probabilitas *crossover*, probabilitas mutasi, dan nilai populasi. Sehingga solusi yang dihasilkan tidak memiliki sifat *completeness* atau tidak menjamin bahwa solusi tersebut merupakan solusi paling optimal dari himpunan solusi yang ada.
2. Penelitian selanjutnya berjudul *Penerapan Metode Greedy Knapsack Dalam Menentukan Komposisi Buah Pada Masalah Keranjang*. Diambil dari *Jurnal Teknologi Informasi*, diteliti oleh Faisal pada 2014 di Jakarta. Penelitian ini membahas bagaimana *knapsack problem* diselesaikan dengan algoritma *greedy*. Algoritma *greedy* memiliki sifat rakus dan tidak memiliki informasi heuristik. Pada penelitian ini cukup berbeda dari penelitian sebelumnya. Jika penelitian sebelumnya mencari *subset*, maka penelitian ini terfokus pada komposisi dari tiga buah. Sehingga bisa dikatakan lebih menyerupai *coin change problem*.

BAB III
METODOLOGI

3.1 Blok Diagram

Berikut adalah diagram blok dari tahapan penelitian ditunjukkan pada gambar 3.1.



Gambar 3.1 *Diagram* Blok

Berikut adalah penjelasan dari setiap tahapan yang ada pada gambar 3.1 yaitu:

3.1.1 Studi Literatur

Penulis mencari judul dan melakukan studi literatur dari penelitian sejenis. Hal ini bertujuan untuk mencari referensi yang berkaitan dengan *knapsack problem*. Sumber didapat dari jurnal, buku, dan situs web.

3.1.2 Perancangan Algoritma

Perancangan algoritma merupakan tahapan desain dari algoritma yang bertujuan untuk mencari metode penyelesaian dari permasalahan yang ada. Di dalam tahapan ini juga akan dilakukan uji coba algoritma untuk melakukan optimasi algoritma. Sehingga algoritma dapat berjalan dengan cepat.

3.1.3 Implementasi Algoritma

Implementasi program merupakan tahapan melakukan pemrograman atau *coding* ke dalam bahasa pemrograman. Sehingga algoritma dapat dipahami oleh mesin atau komputer. Implementasi dilakukan dengan menggunakan bahasa pemrograman dart dan *framework* flutter.

3.1.4 Simulasi Program

Simulasi program merupakan tahapan untuk melakukan uji coba atau *trial and error* pada program yang sedang dikembangkan. Kemudian hasil dianalisis untuk menentukan apakah solusi dapat ditemukan.

3.1.5 Implementasi GUI

Implementasi GUI merupakan tahapan untuk membangun antarmuka pengguna yang menjembatani antara pengguna (*end user)* dengan program inti (*core program*). Sehingga pengguna dapat berinteraksi dengan pengguna awan. GUI juga digunakan untuk meningkatkan pengalaman pengguna atau *user experience*.

3.1.6 Pengambilan Data

Pengambilan data merupakan tahapan untuk melakukan pengukuran terkait hasil akhir dari beberapa studi kasus. Data diambil dari 7 studi kasus yang diharapkan mampu memberikan gambaran bagaimana algoritma BFS bekerja.

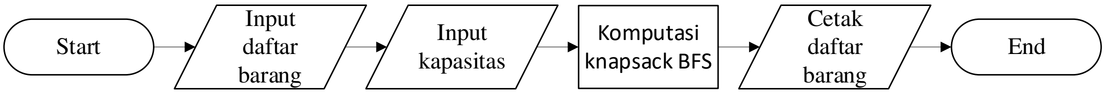
3.1.7 Penyusunan Laporan

Penyusunan laporan merupakan tahapan untuk membuat laporan tugas akhir yang telah dibuat berdasarkan hasil pengambilan data.

3.2 Rancangan Sistem

3.2.1 Gambaran Umum Sistem

Berikut adalah diagram blok terkait gambaran umum sistem yang akan dirancang. Diagram disajikan pada gambar 3.2.

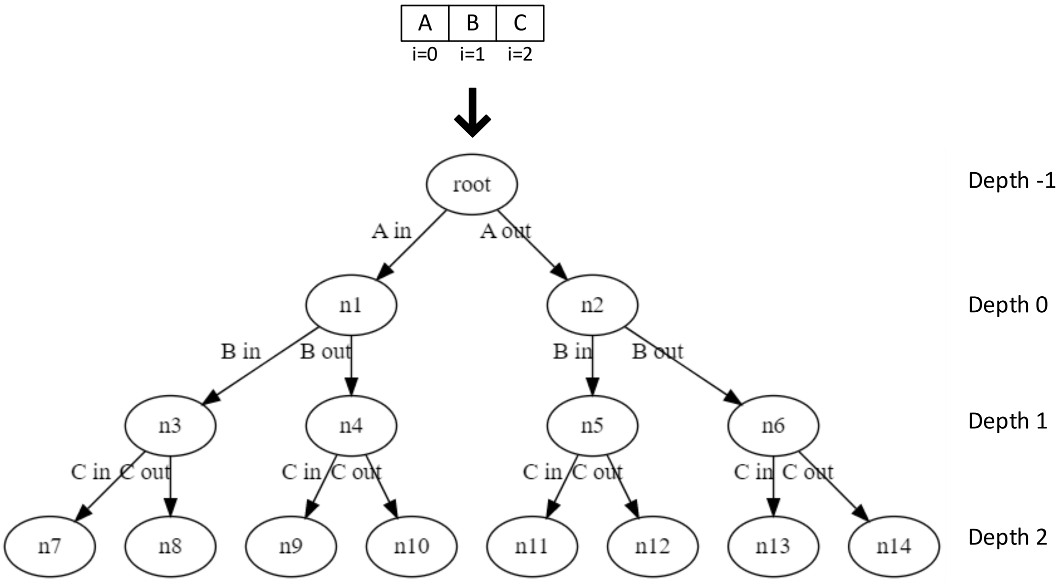


Gambar 3.2 *Flowchart* alur kerja sistem

Pada gambar 3.2 menjelaskan bahwa pengguna akan diminta untuk melakukan entri beberapa informasi barang berupa berat w dan keuntungan p. Kemudian pengguna juga akan diminta untuk menginputkan kapasitas maksimal dari keranjang. Setelah itu sistem akan membangun *tree* dan *queue* untuk melakukan komputasi penjelajahan graf dengan metode BFS. Simpul daun dengan profit tertinggi merupakan tujuan dari pencarian BFS. Apabila simpul ini ditemukan, rute dari simpul daun ke simpul *root* akan ditandai sebagai solusi dari *knapsack problem*. Hasil dari perhitungan kemudian akan ditampilkan kepada pengguna.

3.2.2 Graf dan Matriks

Graf pada penelitian ini tidak direpresentasikan ke dalam matriks ketetanggaan maupun *list* ketetanggaan. Namun direpresentasikan ke dalam *list array* dengan indeks $0 \leq i \leq n$ dan digunakan untuk menunjukkan kedalaman dari *tree*.



Gambar 3.3 Konversi *list array* menjadi *perfect tree*

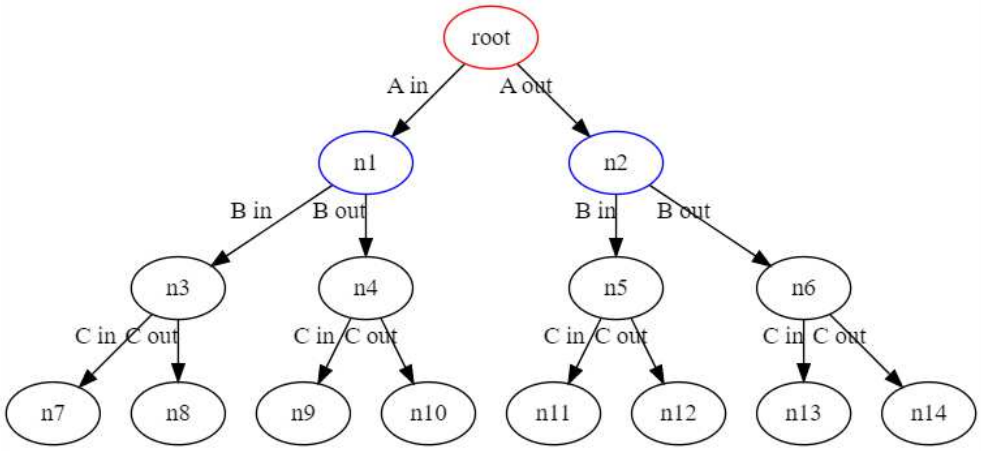
Misalnya pada gambar di atas *array* dengan 3 elemen berupa A, B, dan C merepresentasikan graf dengan kedalaman 4 dan 8 total simpul daun. *Tree* akan dibangun ketika penjelajahan dengan pemodelan *linked object*.

3.2.3 *Decision Tree* BFS

Setiap simpul selain simpul daun memiliki dua cabang yang menunjukkan keputusan antara memilih memasukkan barang ke keranjang atau tidak memasukkan barang ke keranjang. Keputusan ini dapat dilihat pada penulisan angka 0 dan 1 pada nama simpul yang merujuk pada posisi 0 dan 1 pada 0/1 *knapsack problem*. Angka 1 berarti dimasukkan ke keranjang dan angka 0 berarti kebalikannya.

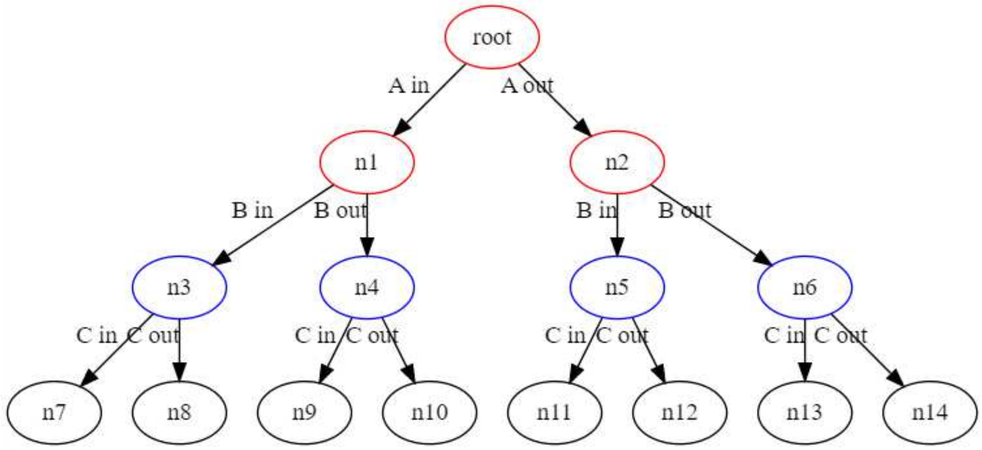
3.2.4 Proses Penjelajahan Graf

Proses penjelajahan graf menggunakan metode BFS (*Breadth-First Search*) dilakukan secara bertahap dari lapisan teratas hingga lapisan terbawah. Apabila *initial state* adalah simpul *root* dan *goal state* adalah simpul n9 dengan asumsi n9 memiliki nilai profit tertinggi. Maka algoritma BFS akan mengunjungi simpul *root* yang berada pada *depth* -1 dan memasukkan simpul n1 dan n2 ke dalam *queue*. Pada kedalaman ini, isi dari *queue* adalah {n1, n2}.



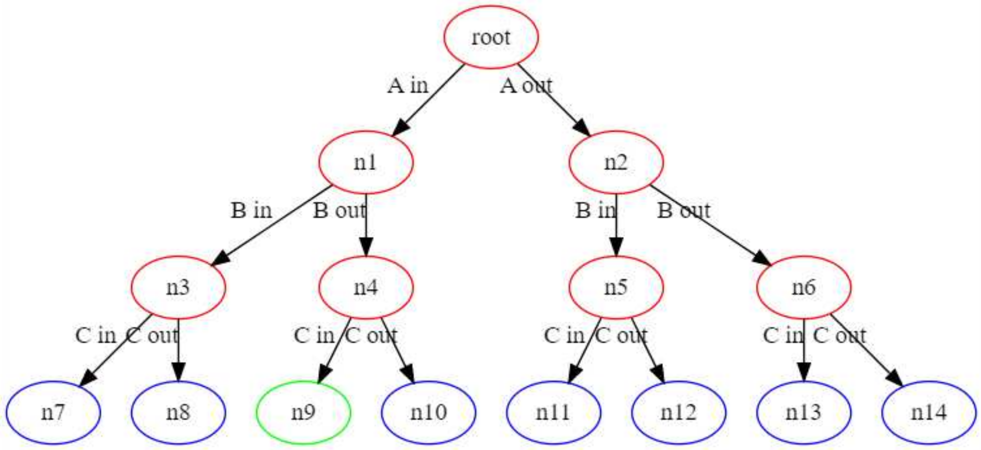
Gambar 3.4 Tahap 1 penjelajahan BFS

Setelah simpul pada *depth* -1 dikunjungi, pencarian berpindah ke simpul yang berada di lapisan bawahnya yaitu *depth* 0. Proses dimulai adalah dengan mengunjungi simpul yang berada pada *queue* yaitu n1 dan memasukkan cabangnya yaitu n3 dan n4 ke dalam *queue*. Kemudian mengunjungi n2 dan cabangnya yaitu n5 dan n6 dimasukkan ke dalam *queue*. Pada kedalaman ini, isi dari *queue* adalah {n3, n4, n5, n6}.



Gambar 3.5 Tahap 2 penjelajahan BFS

Setelah simpul pada *depth* 0 dikunjungi, pencarian berpindah ke simpul yang berada di lapisan bawahnya yaitu *depth* 1. Proses dimulai adalah dengan mengunjungi simpul yang berada pada *queue* yaitu n3 dan memasukkan cabangnya yaitu n7 dan n8 ke dalam *queue*. Kemudian mengunjungi n4 dan cabangnya yaitu n9 dan n10 dimasukkan ke dalam *queue*. Dilanjutkan dengan mengunjungi n5 dan cabangnya yaitu n11 dan n12 dimasukkan ke dalam *queue*. Setelah itu mengunjungi n6 dan cabangnya yaitu n13 dan n13 dimasukkan ke dalam *queue*. Pada kedalaman ini, isi dari *queue* adalah {n7, n8, n9, n10, n11, n12, n13, n14}.



Gambar 3.6 Tahap 3 penjelajahan BFS

Penjelajahan diakhiri dengan menjelajahi isi dari *queue* yaitu n7, namun n7 tidak memiliki cabang. Kemudian dilanjutkan ke n8 dan tidak memiliki cabang juga. Akhirnya n9 dijelajahi dan ditemukan. Proses pencarian berakhir karena n9 telah ditemukan. Pada praktiknya penjelajahan akan tetap diteruskan hingga n14. Oleh karena itu algoritma ini dapat mendapatkan solusi terbaik dari himpunan solusi.

BAB IV
HASIL DAN PEMBAHASAN

Hasil yang telah diperoleh setelah melakukan implementasi berdasarkan metodologi yang disusun serta perancangan sistem disajikan dalam cuplikan layar dan analisis keluaran dari program yang dibangun.

4.1 Proses Perhitungan dengan 7 Kondisi

Untuk menguji coba sistem yang telah dibuat dibutuhkan berbagai macam studi kasus yang merepresentasikan berbagai macam kondisi pada aplikasi dunia nyata. Oleh karena itu, telah dibuat 7 studi kasus dan dirangkum keseluruhan *dataset* ke dalam tabel berikut.

Nama Buah	Berat (g)	Keuntungan (Rp)
Anggur	100	8000
Apel	200	6900
Blimbing	400	6200
Delima	800	5500
Jambu	600	11500
Jeruk	250	3800
Kelengkeng	750	13300
Kiwi	500	18000
Kurma	800	12000
Lemon	450	6000
Mangga	750	10000
Markisa	600	5000
Nanas	800	8000
Buah Naga	900	10000
Pisang	700	12000
Rambutan	500	8900
Salak	850	20000
Srikaya	500	3500
Strawberry	500	14500

Tabel 4.1 Keseluruhan *dataset* uji coba

Berikut adalah informasi lengkap dari 7 *dataset* beserta kapasitas maksimum *knapsack* yang akan digunakan untuk menguji coba sistem pemilihan buah menggunakan algoritma BFS.

1. Tabel *Dataset* 1

   Kapasitas keranjang = 1500

Nama	Weight	Profit
Jambu	600	11500

Kiwi	500	18000
Lemon	450	6000
Buah Naga	400	6200
Apel	200	6900

Tabel 4.2 *Dataset* 1

2. Tabel *Dataset* 2

   Kapasitas keranjang = 1750

Nama	Weight	Profit
Delima	800	5500
Salak	850	20000
Buah Naga	400	6200
Srikaya	500	3500
Markisa	600	5000

Tabel 4.3 *Dataset* 2

3. Tabel *Dataset* 3

   Kapasitas keranjang = 1600

Nama	Weight	Profit
Salak	850	20000
Delima	800	5500
Buah Naga	400	6200
Nanas	800	8000
Mangga	750	10000
Blimbing	400	6200

Tabel 4.4 *Dataset* 3

4. Tabel *Dataset* 4

   Kapasitas keranjang = 1400

Nama	Weight	Profit
Buah Naga	400	6200
Anggur	100	8000
Pisang	700	12000
Kurma	800	12000
Jeruk	250	3800
Kelengkeng	750	13300

Tabel 4.5 *Dataset* 4

5. Tabel *Dataset* 5

   Kapasitas keranjang = 2000

Nama	Weight	Profit
Blimbing	400	6200
Strawberry	500	14500

Jambu	600	11500
Anggur	100	8000
Salak	850	20000
Nanas	800	8000

Tabel 4.6 *Dataset* 5

6. Tabel *Dataset* 6

   Kapasitas keranjang = 1000

Nama	Weight	Profit
Jambu	600	11500
Kurma	800	12000
Nanas	800	8000
Apel	200	6900

Tabel 4.7 *Dataset* 6

7. Tabel *Dataset* 7

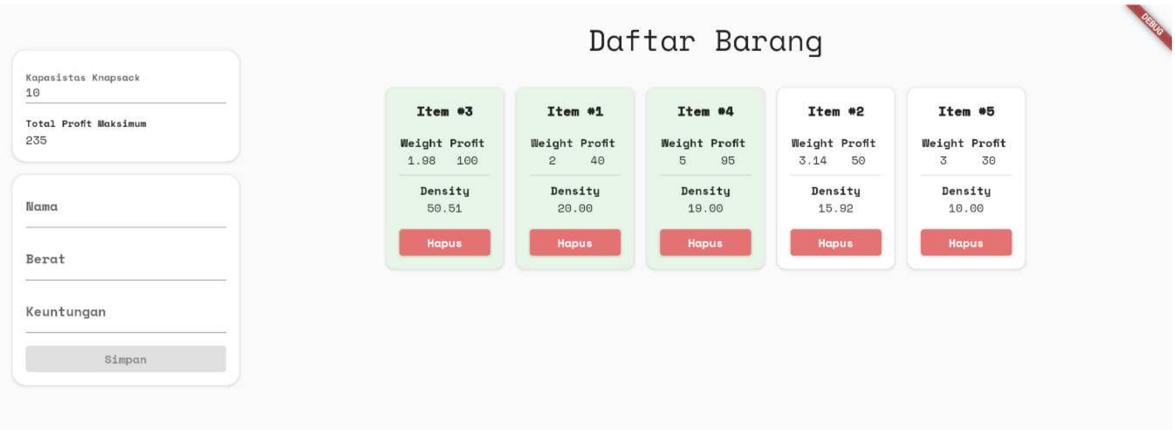
   Kapasitas keranjang = 1500

Nama	Weight	Profit
Blimbing	400	6200
Rambutan	500	8900
Kurma	800	12000
Jeruk	250	3800

Tabel 4.8 *Dataset* 7

4.2 Hasil Perancangan Sistem

Pada saat program dijalankan, akan tampil halaman seperti pada gambar di bawah ini. Sistem ini bersifat reaktif karena menggunakan *framework* dengan *state management* yang mampu mengubah UI ketika terjadi perubahan data.



Kapasistas Knapsack
10

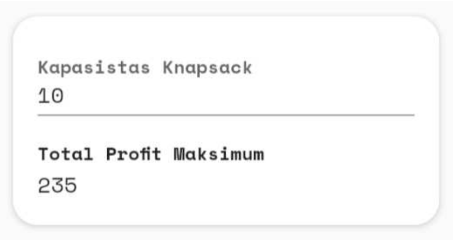
Total Profit Maksimum
235

Daftar Barang

Item #3	Item #1	Item #4	Item #2	Item #5
Weight 1.98 Profit 100	Weight 2 Profit 40	Weight 5 Profit 95	Weight 3.14 Profit 50	Weight 3 Profit 30
Density 50.51	Density 20.00	Density 19.00	Density 15.92	Density 10.00

Gambar 4.1 Halaman utama

Berikut adalah menu yang digunakan untuk meng-*input* nilai kapasitas maksimum dari *knapsack*. Selain itu juga terdapat informasi dari *output* terkait total profit maksimum dari kombinasi pemilihan buah.

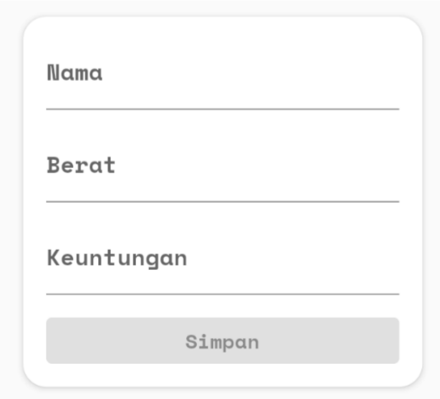


Kapasistas Knapsack
10

Total Profit Maksimum
235

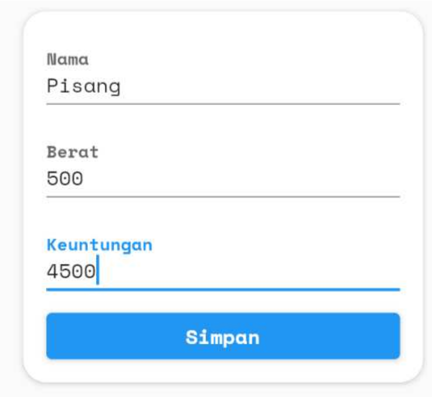
Gambar 4.2 Menu *input* kapasitas *knapsack*

Selain itu juga terdapat menu untuk melakukan *input* informasi barang meliputi nama, nilai berat, dan nilai keuntungan.



Gambar 4.3 Menu *input* barang nonaktif

Ketika semua kolom masih kosong, maka tombol simpan berwarna abu-abu dan tidak dapat diklik. Namun ketika semua kolom terisi, maka tombol simpan akan berwarna biru dan apabila diklik akan menambahkan kartu ke dalam menu daftar barang.



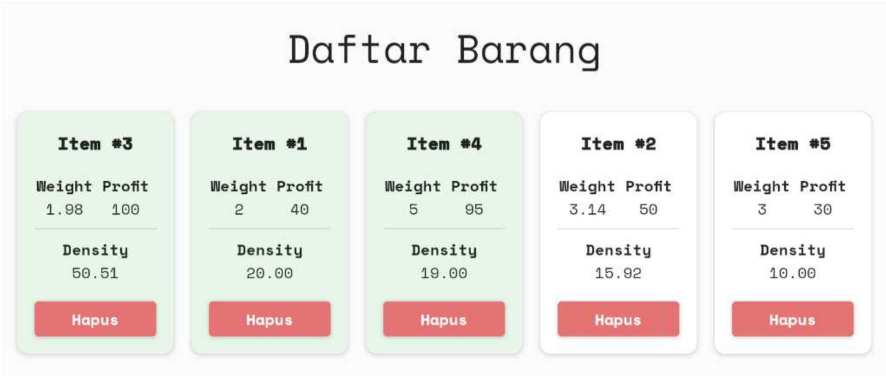
Nama
Pisang

Berat
500

Keuntungan
4500

Gambar 4.4 Menu *input* barang aktif

Pada menu daftar barang di bawah ini menjelaskan bahwa terdapat 5 barang yang telah di *input*. Kartu berwarna hijau menunjukkan bahwa barang tersebut dipilih untuk dimasukkan ke dalam keranjang. Berdasarkan kapasitas *knapsack*, terdapat 3 barang yang dipilih untuk mencapai nilai total profit maksimum. Perubahan ini langsung dapat dilihat setelah melakukan *input* data barang. Tombol hapus digunakan untuk menghapus barang tersebut dari daftar barang.



Daftar Barang

Item #3
Weight Profit
1.98 100
Density
50.51
Hapus

Item #1
Weight Profit
2 40
Density
20.00
Hapus

Item #4
Weight Profit
5 95
Density
19.00
Hapus

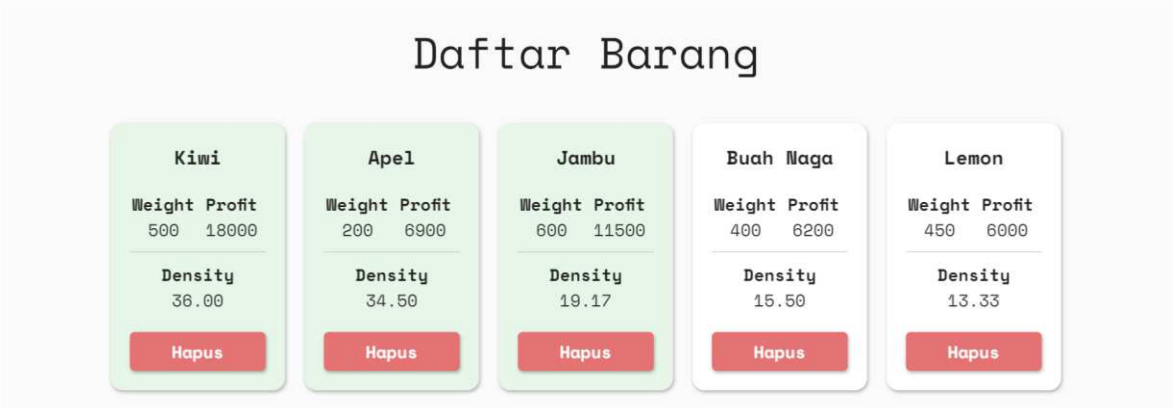
Item #2
Weight Profit
3.14 50
Density
15.92
Hapus

Item #5
Weight Profit
3 30
Density
10.00
Hapus

Gambar 4.5 Menu daftar barang

4.3 Hasil Eksekusi Program

Pada gambar 4.6 di bawah merupakan hasil eksekusi program dari *dataset* 1. Dapat dilihat bahwa total profit dari maksimum dari *subset dataset* 1 adalah 36400 dengan total berat 1300. Buah yang terpilih adalah kiwi, apel, dan jambu. Ketiga buah ini memiliki total keuntungan tertinggi dengan memperhatikan batasan kapasitas *knapsack* yaitu 1500. Sehingga ketiga dari kartu informasi barang berwarna hijau.



Daftar Barang

Kiwi
Weight Profit
500 18000
Density
36.00
Hapus

Apel
Weight Profit
200 6900
Density
34.50
Hapus

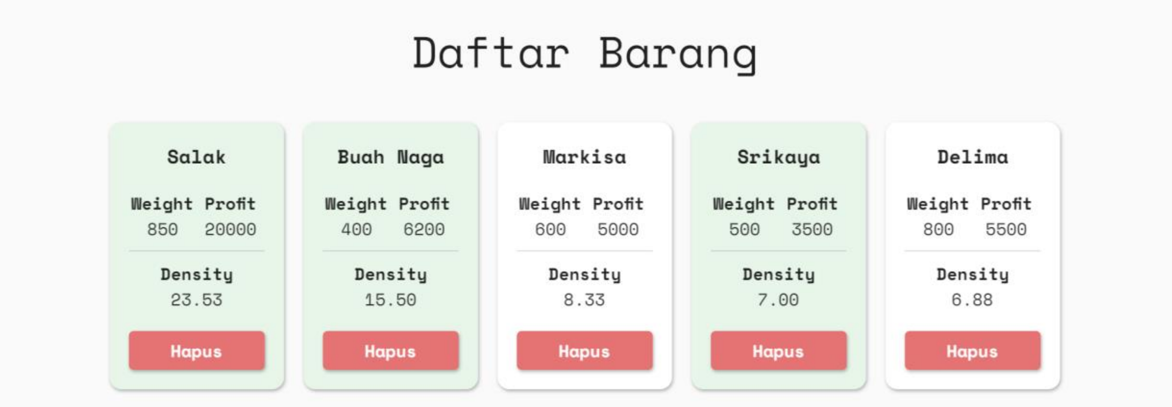
Jambu
Weight Profit
600 11500
Density
19.17
Hapus

Buah Naga
Weight Profit
400 6200
Density
15.50
Hapus

Lemon
Weight Profit
450 6000
Density
13.33
Hapus

Gambar 4.6 Eksekusi program pada *dataset* 1

Sedangkan pada gambar 4.7 merupakan hasil eksekusi program dari *dataset* 2. Hasil yang didapat dari eksekusi program ini adalah dua buah yang terpilih yaitu salak dan buah naga memiliki total profit 29700. Total berat dari kedua buah ini yaitu 1250 tidak melebihi batas maksimum *knapsack* yaitu 1750. Meskipun total berat jauh dari batasan keranjang, kombinasi ini merupakan solusi terbaik.



Daftar Barang

Salak
Weight Profit
850 20000
Density
23.53

Buah Naga
Weight Profit
400 6200
Density
15.50

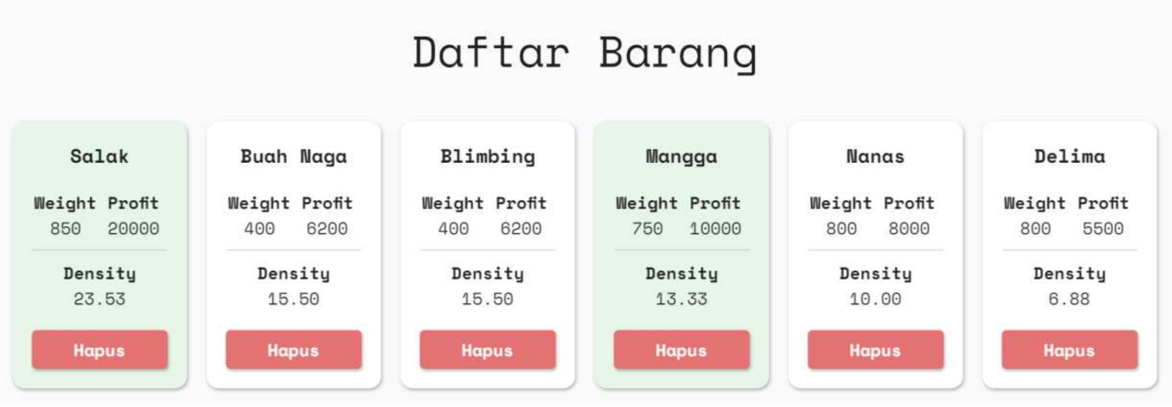
Markisa
Weight Profit
600 5000
Density
8.33

Srikaya
Weight Profit
500 3500
Density
7.00

Delima
Weight Profit
800 5500
Density
6.88

Gambar 4.7 Eksekusi program pada *dataset* 2

Pada *dataset* 3, jumlah total profit yang dicapai adalah 30000 dengan total *weight* tepat 1600. Studi kasus ini menunjukkan bahwa algoritma juga bekerja untuk kombinasi sempurna.



Daftar Barang

Salak
Weight Profit
850 20000
Density
23.53

Buah Naga
Weight Profit
400 6200
Density
15.50

Blimbing
Weight Profit
400 6200
Density
15.50

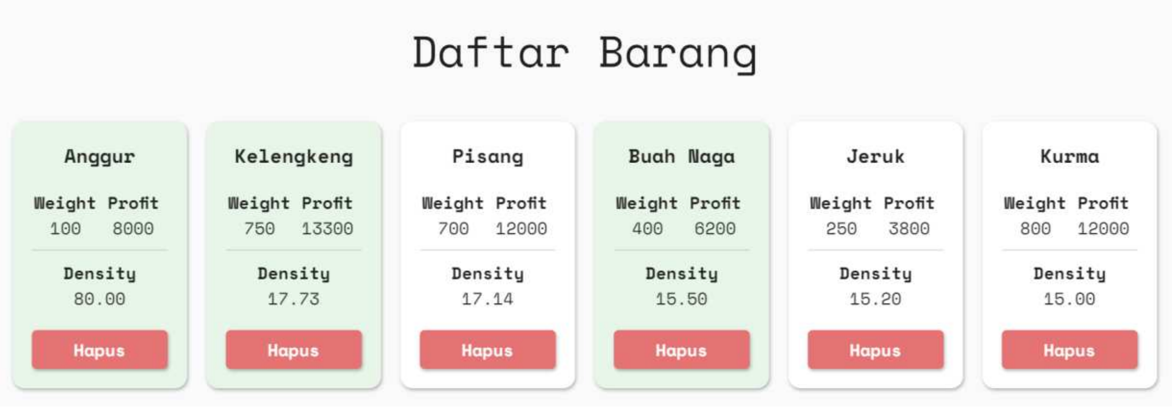
Mangga
Weight Profit
750 10000
Density
13.33

Nanas
Weight Profit
800 8000
Density
10.00

Delima
Weight Profit
800 5500
Density
6.88

Gambar 4.8 Eksekusi program pada *dataset* 3

Gambar 4.9 menunjukkan hasil eksekusi program pada *dataset* 4. Hasil menunjukkan bahwa buah yang terpilih adalah anggur, kelengkeng, dan buah naga. Total keuntungan dari ketiga buah ini adalah 27500.



Daftar Barang

Anggur
Weight Profit
100 8000
Density
80.00

Kelengkeng
Weight Profit
750 13300
Density
17.73

Pisang
Weight Profit
700 12000
Density
17.14

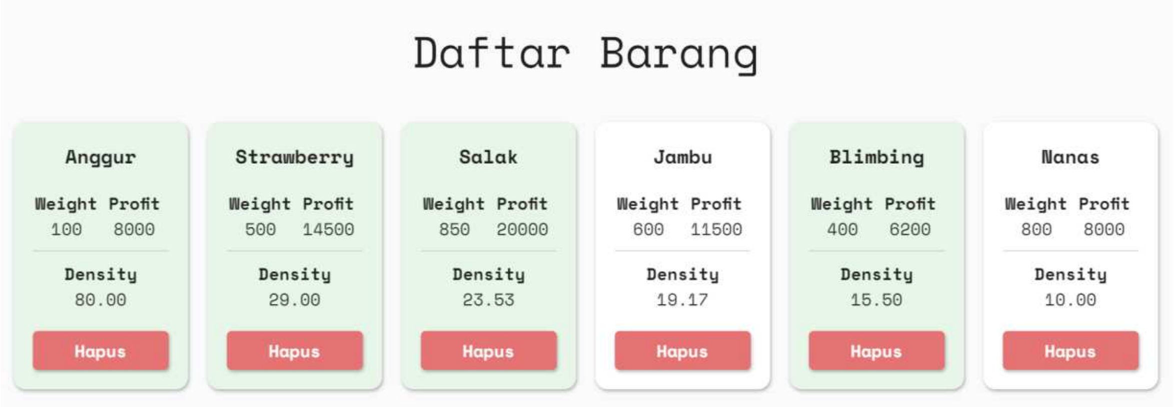
Buah Naga
Weight Profit
400 6200
Density
15.50

Jeruk
Weight Profit
250 3800
Density
15.20

Kurma
Weight Profit
800 12000
Density
15.00

Gambar 4.9 Eksekusi program pada *dataset* 4

Pada gambar 4.10 menunjukkan hasil eksekusi program pada *dataset* 5. Dari 6 buah dipilih 4 buah yang memiliki total profit terbesar yaitu 48700.



Daftar Barang

Anggur
Weight Profit
100 8000
Density
80.00

Strawberry
Weight Profit
500 14500
Density
29.00

Salak
Weight Profit
850 20000
Density
23.53

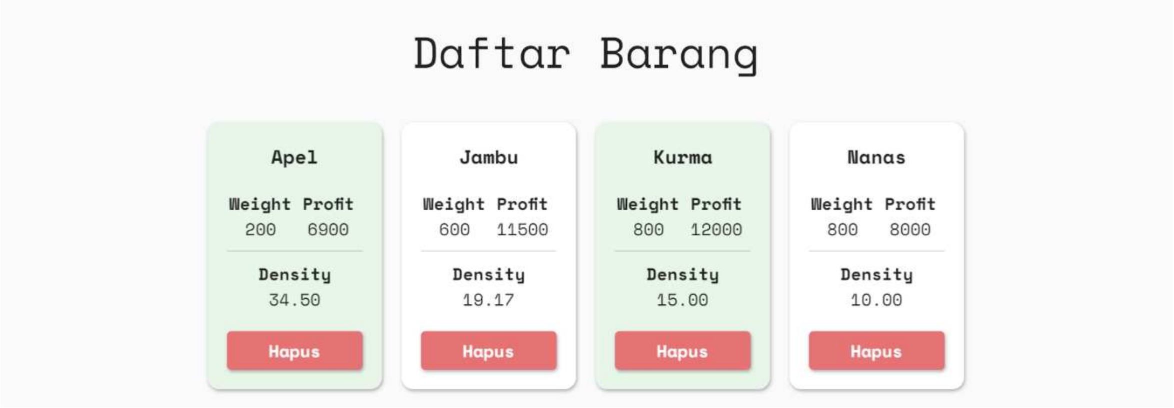
Jambu
Weight Profit
600 11500
Density
19.17

Blimbing
Weight Profit
400 6200
Density
15.50

Nanas
Weight Profit
800 8000
Density
10.00

Gambar 4.10 Eksekusi program pada *dataset* 5

Sedangkan pada gambar 4.11 yang menunjukkan hasil eksekusi program pada *dataset* 6 didapat 2 buah dari 4 buah yang di-*input*. Total profit adalah 18900 dan total berat adalah 1000.



Daftar Barang

Apel
Weight Profit
200 6900
Density
34.50

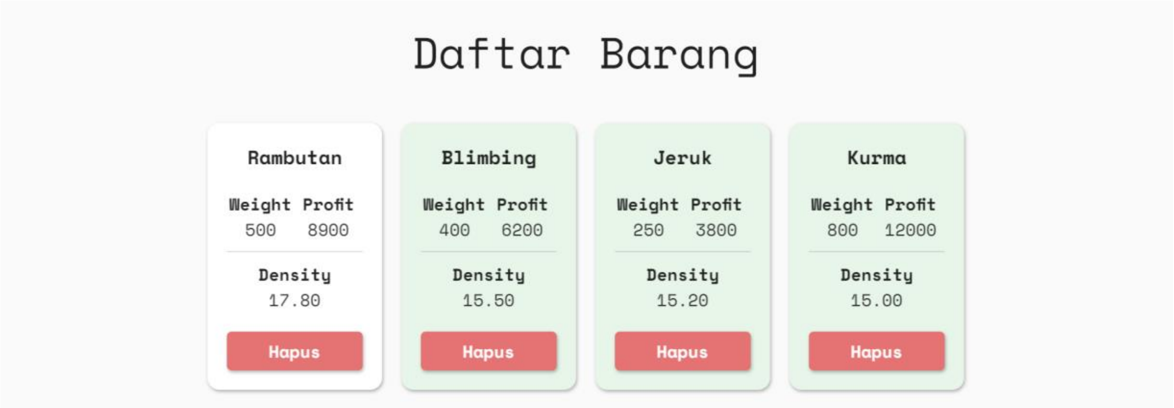
Jambu
Weight Profit
600 11500
Density
19.17

Kurma
Weight Profit
800 12000
Density
15.00

Nanas
Weight Profit
800 8000
Density
10.00

Gambar 4.11 Eksekusi program pada *dataset* 6

Pada gambar 4.12 yang menunjukkan hasil eksekusi program pada *dataset* 7 didapat 3 buah dari 4 buah yang di-*input*. Total profit adalah 22000 dan total berat adalah 1450.



Daftar Barang

Rambutan
Weight Profit
500 8900
Density
17.80

Blimbing
Weight Profit
400 6200
Density
15.50

Jeruk
Weight Profit
250 3800
Density
15.20

Kurma
Weight Profit
800 12000
Density
15.00

Gambar 4.12 Eksekusi program pada *dataset* 7

BAB V
PENUTUP

5.1 Kesimpulan

Berdasarkan *output* dari sistem yang telah dibangun mengenai *knapsack problem* dengan menggunakan algoritma *brute-force* berbasis *breadth-first search* didapat bahwa metode ini dapat menyelesaikan masalah tersebut dan dapat mencapai solusi dengan sifat *completeness*. Algoritma ini dinilai cukup unggul dibandingkan algoritma *greedy* dan algoritma genetika, meskipun waktu eksekusinya lebih panjang karena memiliki kompleksitas waktu tinggi.

5.2 Saran

Saran yang dapat digunakan untuk pengembangan selanjutnya adalah penambahan fitur lain yang memberikan informasi heuristik kepada obyek sehingga pemilihan buah tidak hanya berdasarkan total profit tapi juga keseimbangan tema kombinasi buah. Misalnya tema buah pada parsel adalah asam sehingga buah yang dipilih cenderung yang bersifat asam.

BAB VI
REFERENSI

Id, F. F. C. (n.d.). *PENERAPAN METODE GREEDY KNAPSACK DALAM MENENTUKAN KOMPOSISI BUAH PADA MASALAH KERANJANG*.

Lumenta, A. (2014). Perbandingan Metode Pencarian Depth-First Search, Breadth-First Search Dan Best-First Search Pada Permainan 8-Puzzle. *Jurusan Teknik Elektro Fakultas Teknik UNSRAT*.

Pisinger, D. (1995). *Algorithms for Knapsack Problems*.

Rahim, R., Abdullah, D., Nurarif, S., Ramadhan, M., Anwar, B., Dahria, M., Nasution, S. D., Diansyah, T. M., & Khairani, M. (2018). Breadth First Search Approach for Shortest Path Solution in Cartesian Area. *Journal of Physics: Conference Series*, *1019*(1). https://doi.org/10.1088/1742-6596/1019/1/012036

Setemen, K. (2010). IMPLEMENTASI ALGORITMA GENETIKA PADA KNAPSACK PROBLEM UNTUK OPTIMASI PEMILIHAN BUAH KEMASAN KOTAK. In *Seminar Nasional Aplikasi Teknologi Informasi*.

Supriana, I. W. (2016). Optimalisasi Penyelesaian Knapsack Problem Dengan Algoritma Genetika. *Lontar Komputer : Jurnal Ilmiah Teknologi Informasi*, 182. https://doi.org/10.24843/lkjiti.2016.v07.i03.p06

LAMPIRAN

Makna kolom status pada tabel iterasi dijelaskan di bawah berikut :

1. **ok** berarti cabang simpul dapat diterima karena total weight lebih rendah daripada kapasitas knapsack.
2. **overload** berarti cabang simpul tidak dapat diterima karena total weight lebih tinggi daripada kapasitas knapsack. Sehingga cabang dari cabang ini tidak akan dihitung karena tidak menuju kepada solusi.
3. **dead path** berarti cabang simpul tidak menuju ke solusi karena hasilnya sama saja dengan simpul parent.
4. Strip "–" berarti simpul ini tidak memiliki cabang.

A. Perhitungan *dataset* 1

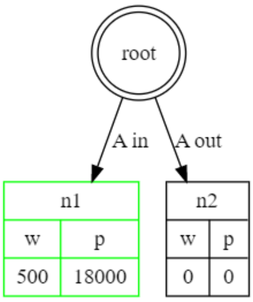
Kapasitas keranjang = 1500

Nama	Weight	Profit	Density
Jambu	600.0	11500.0	~19.16
Kiwi	500.0	18000.0	36.0
Lemon	450.0	6000.0	~13.33
Buah Naga	400.0	6200.0	15.5
Apel	200.0	6900.0	34.5

Lakukan pengurutan berdasarkan *density* secara *descending* untuk mempermudah pencarian berdasarkan *greedy by density*.

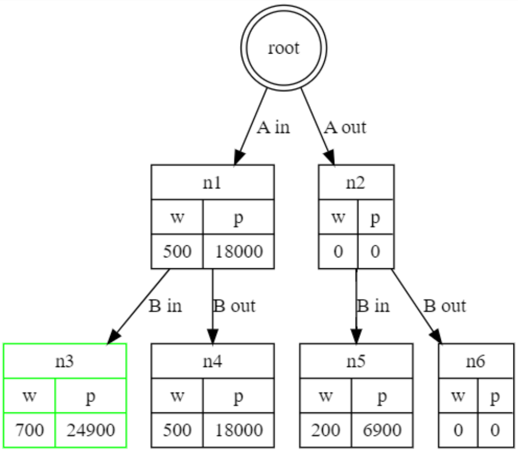
Label	Nama	Weight	Profit	Density
A	Kiwi	500.0	18000.0	36.0
B	Apel	200.0	6900.0	34.5
C	Jambu	600.0	11500.0	~19.16
D	Buah Naga	400.0	6200.0	15.5
E	Lemon	450.0	6000.0	~13.33

Langkah pada *depth* 0



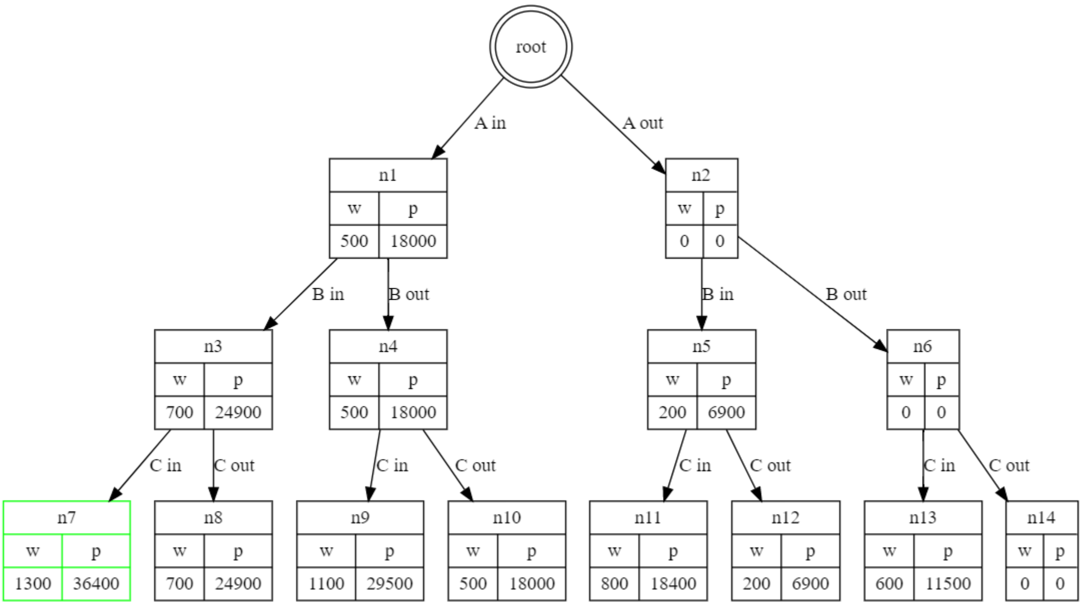
itr	current	branch	status	solution	queue
1	root	n1 (A in)	ok	n1	{n1}
2	root	n2 (A out)	ok	n1	{n1,n2}

Langkah pada *depth* 1



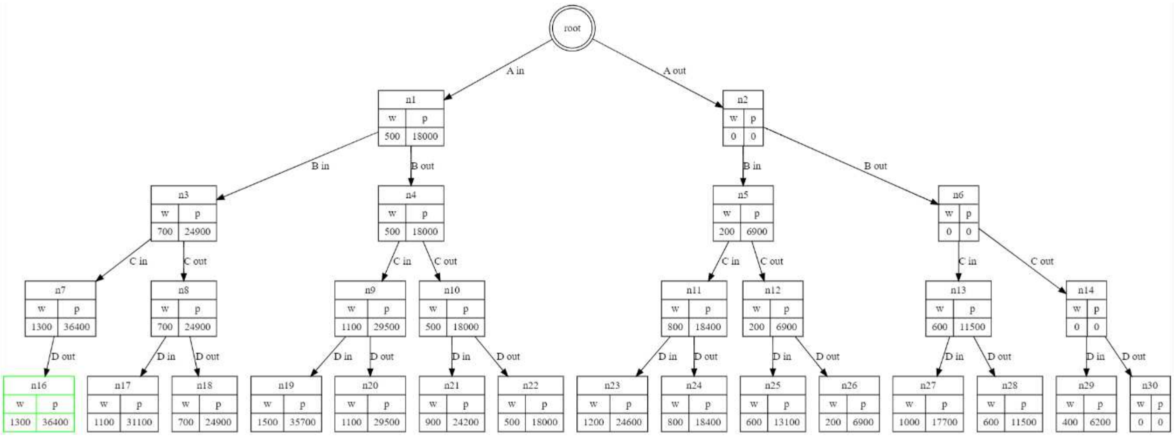
itr	current	branch	status	solution	queue
3	n1	n3 (B in)	ok	n3	{n2,n3}
4	n1	n4 (B out)	ok	n3	{n2,n3,n4}
5	n2	n5 (B in)	ok	n3	{n3,n4,n5}
6	n2	n6 (B out)	ok	n3	{n3,n4,n5,n6}

Langkah pada *depth* 2



itr	current	branch	status	solution	queue
7	n3	n7 (C in)	ok	n7	{n4,n5,n6,n7}
8	n3	n8 (C out)	ok	n7	{n4,n5,n6,n7,n8}
9	n4	n9 (C in)	ok	n7	{n5,n6,n7,n8,n9}
10	n4	n10 (C out)	ok	n7	{n5,n6,n7,n8,n9,n10}
11	n5	n11 (C in)	ok	n7	{n6,n7,n8,n9,n10,n11}
12	n5	n12 (C out)	ok	n7	{n6,n7,n8,n9,n10,n11,n12}
13	n6	n13 (C in)	ok	n7	{n7,n8,n9,n10,n11,n12,n13}
14	n6	n14 (C out)	ok	n7	{n7,n8,n9,n10,n11,n12,n13,n14}

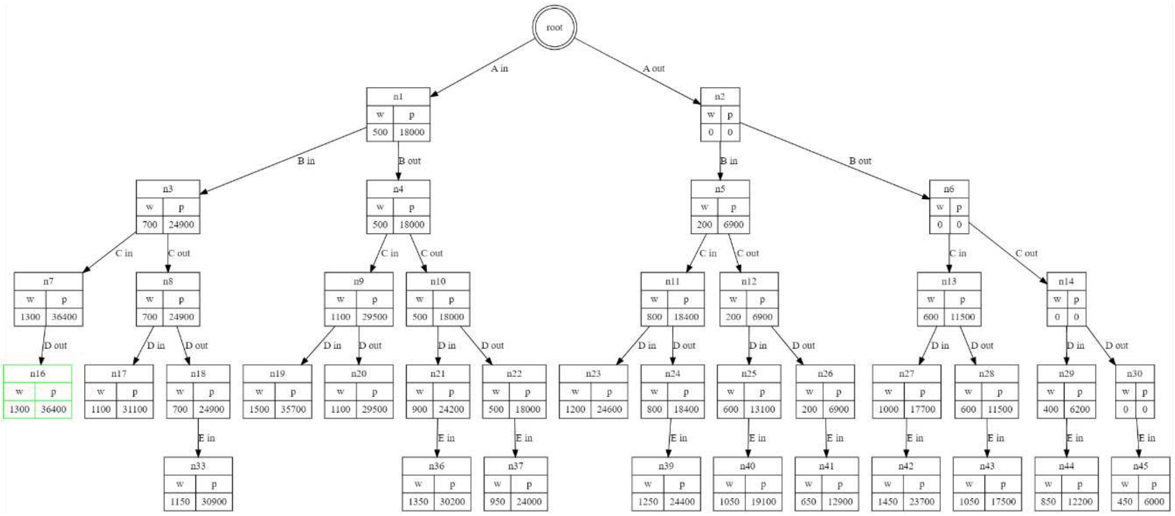
Langkah pada *depth* 3



itr	current	branch	status	solution	queue
15	n7	n15 (D in)	overload	n7	{n8,n9,n10,n11,n12,n13,n14}
16	n7	n16 (D out)	ok	n16	{n8,n9,n10,n11,n12,n13,n14,n16}

17	n8	n17 (D in)	ok	n16	{n9,n10,n11,n12,n13,n14,n16,n17}
18	n8	n18 (D out)	ok	n16	{n9,n10,n11,n12,n13,n14,n16,n17,n18}
19	n9	n19 (D in)	ok	n16	{n10,n11,n12,n13,n14,n16,n17,n18,n19}
20	n9	n20 (D out)	ok	n16	{n10,n11,n12,n13,n14,n16,n17,n18,n19,n20}
21	n10	n21 (D in)	ok	n16	{n11,n12,n13,n14,n16,n17,n18,n19,n20,n21}
22	n10	n22 (D out)	ok	n16	{n11,n12,n13,n14,n16,n17,n18,n19,n20,n21,n22}
23	n11	n23 (D in)	ok	n16	{n12,n13,n14,n16,n17,n18,n19,n20,n21,n22,n23}
24	n11	n24 (D out)	ok	n16	{n12,n13,n14,n16,n17,n18,n19,n20,n21,n22,n23,n24}
25	n12	n25 (D in)	ok	n16	{n13,n14,n16,n17,n18,n19,n20,n21,n22,n23,n24,n25}
26	n12	n26 (D out)	ok	n16	{n13,n14,n16,n17,n18,n19,n20,n21,n22,n23,n24,n25,n26}
27	n13	n27 (D in)	ok	n16	{n14,n16,n17,n18,n19,n20,n21,n22,n23,n24,n25,n26,n27}
28	n13	n28 (D out)	ok	n16	{n14,n16,n17,n18,n19,n20,n21,n22,n23,n24,n25,n26,n27,n28}
29	n14	n29 (D in)	ok	n16	{n16,n17,n18,n19,n20,n21,n22,n23,n24,n25,n26,n27,n28,n29}
30	n14	n30 (D out)	ok	n16	{n16,n17,n18,n19,n20,n21,n22,n23,n24,n25,n26,n27,n28,n29,n30}

Langkah pada *depth* 4



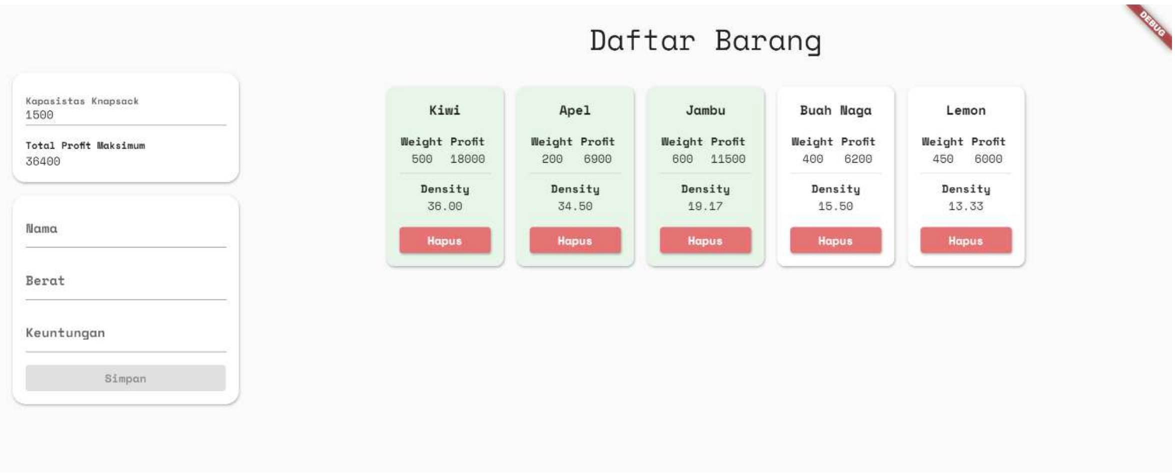
itr	current	branch	status	solution	queue
31	n16	n31 (E in)	overload	n16	{n17,n18,n19,n20,n21,n22,n23,n24,n25,n26,n27,n28,n29,n30}
32	n16	n30 (D out)	dead path	n16	{n17,n18,n19,n20,n21,n22,n23,n24,n25,n26,n27,n28,n29,n30}
33	n17	n32 (E in)	overload	n16	{n18,n19,n20,n21,n22,n23,n24,n25,n26,n27,n28,n29,n30}
34	n17	n30 (D out)	dead path	n16	{n18,n19,n20,n21,n22,n23,n24,n25,n26,n27,n28,n29,n30}
35	n18	n33 (E in)	ok	n16	{n19,n20,n21,n22,n23,n24,n25,n26,n27,n28,n29,n30,n33}
36	n18	n30 (D out)	dead path	n16	{n19,n20,n21,n22,n23,n24,n25,n26,n27,n28,n29,n30,n33}
37	n19	n34 (E in)	overload	n16	{n20,n21,n22,n23,n24,n25,n26,n27,n28,n29,n30,n33}
38	n19	n30 (D out)	dead path	n16	{n20,n21,n22,n23,n24,n25,n26,n27,n28,n29,n30,n33}
39	n20	n35 (E in)	overload	n16	{n21,n22,n23,n24,n25,n26,n27,n28,n29,n30,n33}
40	n20	n30 (D out)	dead path	n16	{n21,n22,n23,n24,n25,n26,n27,n28,n29,n30,n33}
41	n21	n36 (E in)	ok	n16	{n22,n23,n24,n25,n26,n27,n28,n29,n30,n33,n36}

42	n21	n30 (D out)	dead path	n16	{n22,n23,n24,n25,n26,n27,n28,n29,n30,n33,n36}
43	n22	n37 (E in)	ok	n16	{n23,n24,n25,n26,n27,n28,n29,n30,n33,n36,n37}
44	n22	n30 (D out)	dead path	n16	{n23,n24,n25,n26,n27,n28,n29,n30,n33,n36,n37}
45	n23	n38 (E in)	overload	n16	{n24,n25,n26,n27,n28,n29,n30,n33,n36,n37}
46	n23	n30 (D out)	dead path	n16	{n24,n25,n26,n27,n28,n29,n30,n33,n36,n37}
47	n24	n39 (E in)	ok	n16	{n25,n26,n27,n28,n29,n30,n33,n36,n37,n39}
48	n24	n30 (D out)	dead path	n16	{n25,n26,n27,n28,n29,n30,n33,n36,n37,n39}
49	n25	n40 (E in)	ok	n16	{n26,n27,n28,n29,n30,n33,n36,n37,n39,n40}
50	n25	n30 (D out)	dead path	n16	{n26,n27,n28,n29,n30,n33,n36,n37,n39,n40}
51	n26	n41 (E in)	ok	n16	{n27,n28,n29,n30,n33,n36,n37,n39,n40,n41}
52	n26	n30 (D out)	dead path	n16	{n27,n28,n29,n30,n33,n36,n37,n39,n40,n41}
53	n27	n42 (E in)	ok	n16	{n28,n29,n30,n33,n36,n37,n39,n40,n41,n42}
54	n27	n30 (D out)	dead path	n16	{n28,n29,n30,n33,n36,n37,n39,n40,n41,n42}
55	n28	n43 (E in)	ok	n16	{n29,n30,n33,n36,n37,n39,n40,n41,n42,n43}
56	n28	n30 (D out)	dead path	n16	{n29,n30,n33,n36,n37,n39,n40,n41,n42,n43}
57	n29	n44 (E in)	ok	n16	{n30,n33,n36,n37,n39,n40,n41,n42,n43,n44}

58	n29	n30 (D out)	dead path	n16	{n30,n33,n36,n37,n39,n40,n41,n42,n43,n44}
59	n30	n45 (E in)	ok	n16	{n33,n36,n37,n39,n40,n41,n42,n43,n44,n45}
60	n30	n30 (D out)	dead path	n16	{n33,n36,n37,n39,n40,n41,n42,n43,n44,n45}
61	n33	-	-	n16	{n36,n37,n39,n40,n41,n42,n43,n44,n45}
62	n36	-	-	n16	{n37,n39,n40,n41,n42,n43,n44,n45}
63	n37	-	-	n16	{n39,n40,n41,n42,n43,n44,n45}
64	n39	-	-	n16	{n40,n41,n42,n43,n44,n45}
65	n40	-	-	n16	{n41,n42,n43,n44,n45}
66	n41	-	-	n16	{n42,n43,n44,n45}
67	n42	-	-	n16	{n43,n44,n45}
68	n43	-	-	n16	{n44,n45}
69	n44	-	-	n16	{n45}
70	n45	-	-	n16	{}

Solusi berakhir pada simpul yang memiliki total profit tertinggi yaitu n16 dengan nilai total profit 36400. Sehingga didapatkan hasil sebagai berikut.

Label	**Nama**	**Weight**	**Profit**	**Density**	**Ambil**
A	Kiwi	500.0	18000.0	36.0	Ya
B	Apel	200.0	6900.0	34.5	Ya
C	Jambu	600.0	11500.0	~19.16	Ya
D	Buah Naga	400.0	6200.0	15.5	Tidak
E	Lemon	450.0	6000.0	~13.33	Tidak



Daftar Barang

Kapasistas Knapsack
1500

Total Profit Maksimum
36400

Nama

Berat

Keuntungan

Kiwi
Weight Profit
500 18000
Density
36.00

Apel
Weight Profit
200 6900
Density
34.50

Jambu
Weight Profit
600 11500
Density
19.17

Buah Naga
Weight Profit
400 6200
Density
15.50

Lemon
Weight Profit
450 6000
Density
13.33

B. Perhitungan dataset 2

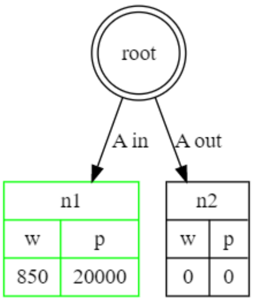
Kapasitas keranjang = 1750

Nama	Weight	Profit	Density
Delima	800.0	5500.0	6.875
Salak	850.0	20000.0	~23.52
Buah Naga	400.0	6200.0	15.5
Srikaya	500.0	3500.0	7.0
Markisa	600.0	5000.0	~8.33

Lakukan pengurutan berdasarkan *density* secara *descending* untuk mempermudah pencarian berdasarkan *greedy by density*.

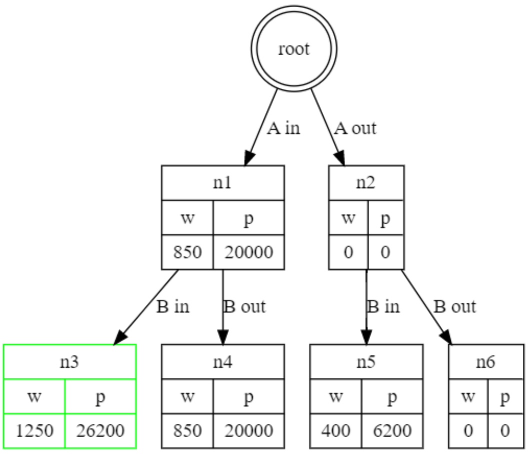
Label	Nama	Weight	Profit	Density
A	Salak	850.0	20000.0	~23.52
B	Buah Naga	400.0	6200.0	15.5
C	Markisa	600.0	5000.0	~8.333333333333334
D	Srikaya	500.0	3500.0	7.0
E	Delima	800.0	5500.0	6.875

Langkah pada *depth* 0



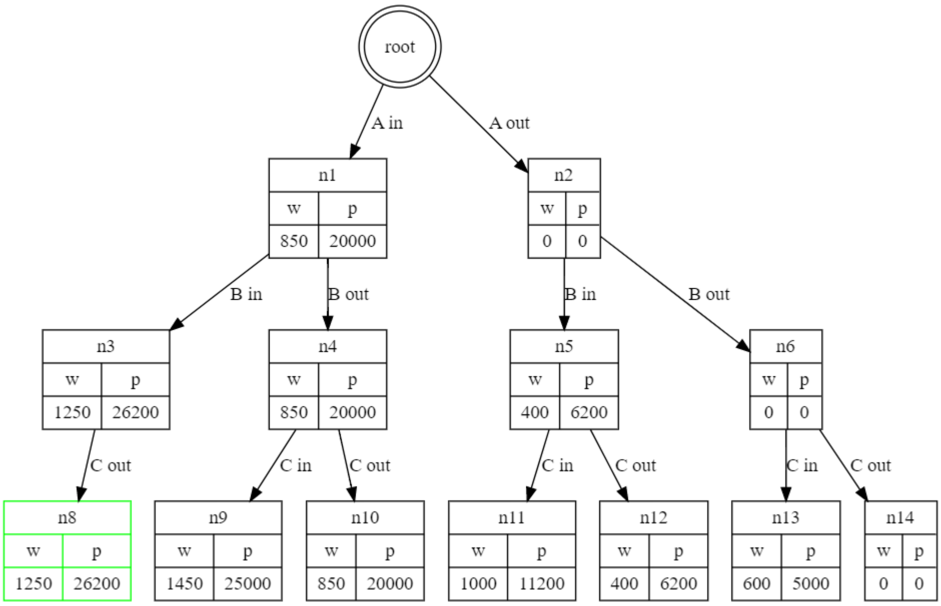
itr	current	branch	status	solution	queue
1	root	n1 (A in)	ok	n1	{n1}
2	root	n2 (A out)	ok	n1	{n1,n2}

Langkah pada *depth* 1



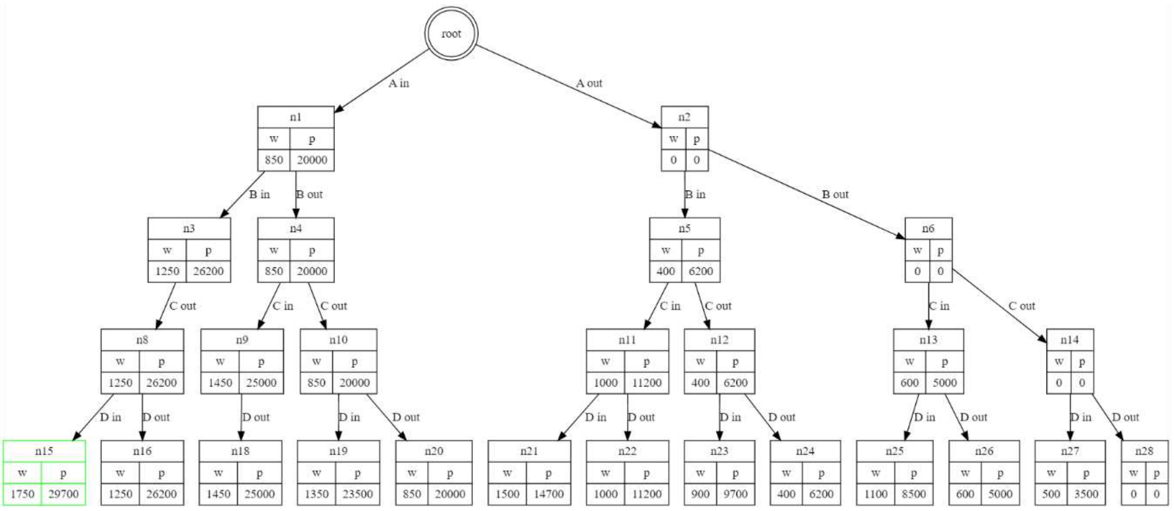
itr	current	branch	status	solution	queue
3	n1	n3 (B in)	ok	n3	{n2,n3}
4	n1	n4 (B out)	ok	n3	{n2,n3,n4}
5	n2	n5 (B in)	ok	n3	{n3,n4,n5}
6	n2	n6 (B out)	ok	n3	{n3,n4,n5,n6}

Langkah pada *depth* 2



itr	current	branch	status	solution	queue
7	n3	n7 (C in)	overload	n3	{n4,n5,n6}
8	n3	n8 (C out)	ok	n8	{n4,n5,n6,n8}
9	n4	n9 (C in)	ok	n8	{n5,n6,n8,n9}
10	n4	n10 (C out)	ok	n8	{n5,n6,n8,n9,n10}
11	n5	n11 (C in)	ok	n8	{n6,n8,n9,n10,n11}
12	n5	n12 (C out)	ok	n8	{n6,n8,n9,n10,n11,n12}
13	n6	n13 (C in)	ok	n8	{n8,n9,n10,n11,n12,n13}
14	n6	n14 (C out)	ok	n8	{n8,n9,n10,n11,n12,n13,n14}

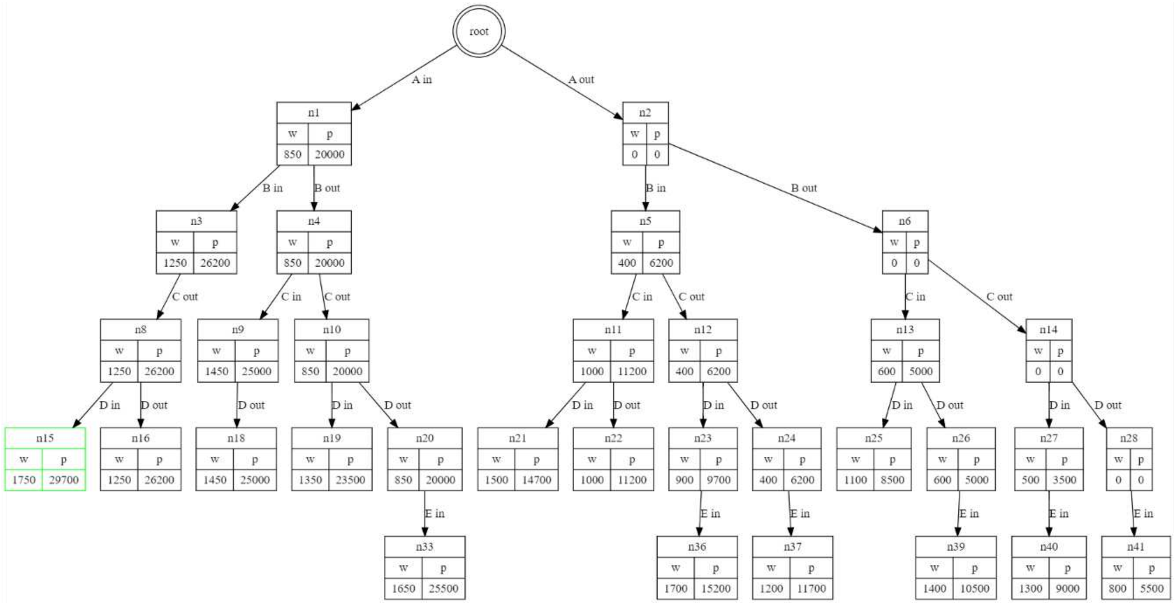
Langkah pada *depth* 3



itr	current	branch	status	solution	queue

15	n8	n15 (D in)	ok	n15	{n9,n10,n11,n12,n13,n14,n15}
16	n8	n16 (D out)	ok	n15	{n9,n10,n11,n12,n13,n14,n15,n16}
17	n9	n17 (D in)	overload	n15	{n10,n11,n12,n13,n14,n15,n16}
18	n9	n18 (D out)	ok	n15	{n10,n11,n12,n13,n14,n15,n16,n18}
19	n10	n19 (D in)	ok	n15	{n11,n12,n13,n14,n15,n16,n18,n19}
20	n10	n20 (D out)	ok	n15	{n11,n12,n13,n14,n15,n16,n18,n19,n20}
21	n11	n21 (D in)	ok	n15	{n12,n13,n14,n15,n16,n18,n19,n20,n21}
22	n11	n22 (D out)	ok	n15	{n12,n13,n14,n15,n16,n18,n19,n20,n21,n22}
23	n12	n23 (D in)	ok	n15	{n13,n14,n15,n16,n18,n19,n20,n21,n22,n23}
24	n12	n24 (D out)	ok	n15	{n13,n14,n15,n16,n18,n19,n20,n21,n22,n23,n24}
25	n13	n25 (D in)	ok	n15	{n14,n15,n16,n18,n19,n20,n21,n22,n23,n24,n25}
26	n13	n26 (D out)	ok	n15	{n14,n15,n16,n18,n19,n20,n21,n22,n23,n24,n25,n26}
27	n14	n27 (D in)	ok	n15	{n15,n16,n18,n19,n20,n21,n22,n23,n24,n25,n26,n27}
28	n14	n28 (D out)	ok	n15	{n15,n16,n18,n19,n20,n21,n22,n23,n24,n25,n26,n27,n28}

Langkah pada *depth* 4

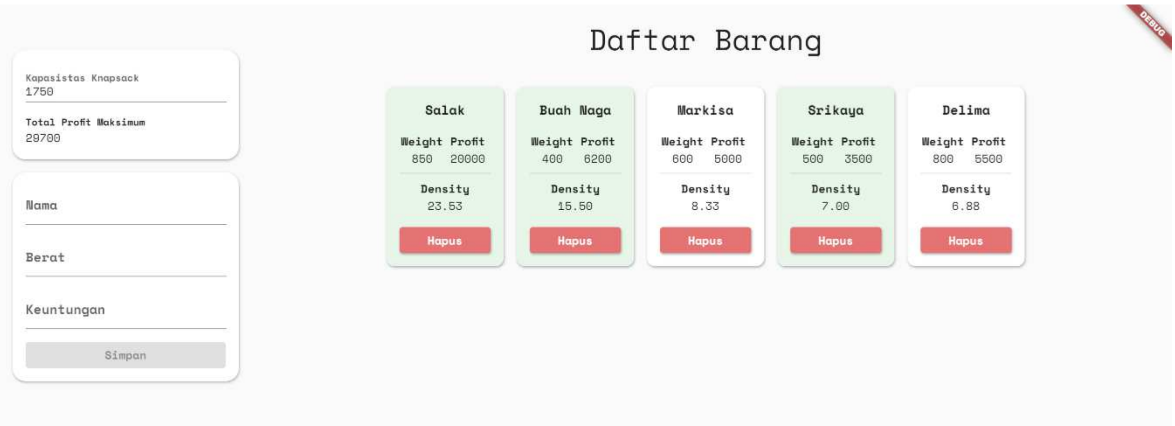


itr	current	branch	status	solution	queue
29	n15	n29 (E in)	overload	n15	{n16,n18,n19,n20,n21,n22,n23,n24,n25,n26,n27,n28}
30	n15	n28 (D out)	dead path	n15	{n16,n18,n19,n20,n21,n22,n23,n24,n25,n26,n27,n28}
31	n16	n30 (E in)	overload	n15	{n18,n19,n20,n21,n22,n23,n24,n25,n26,n27,n28}
32	n16	n28 (D out)	dead path	n15	{n18,n19,n20,n21,n22,n23,n24,n25,n26,n27,n28}
33	n18	n31 (E in)	overload	n15	{n19,n20,n21,n22,n23,n24,n25,n26,n27,n28}
34	n18	n28 (D out)	dead path	n15	{n19,n20,n21,n22,n23,n24,n25,n26,n27,n28}
35	n19	n32 (E in)	overload	n15	{n20,n21,n22,n23,n24,n25,n26,n27,n28}
36	n19	n28 (D out)	dead path	n15	{n20,n21,n22,n23,n24,n25,n26,n27,n28}
37	n20	n33 (E in)	ok	n15	{n21,n22,n23,n24,n25,n26,n27,n28,n33}
38	n20	n28 (D out)	dead path	n15	{n21,n22,n23,n24,n25,n26,n27,n28,n33}
39	n21	n34 (E in)	overload	n15	{n22,n23,n24,n25,n26,n27,n28,n33}

40	n21	n28 (D out)	dead path	n15	{n22,n23,n24,n25,n26,n27,n28,n33}
41	n22	n35 (E in)	overload	n15	{n23,n24,n25,n26,n27,n28,n33}
42	n22	n28 (D out)	dead path	n15	{n23,n24,n25,n26,n27,n28,n33}
43	n23	n36 (E in)	ok	n15	{n24,n25,n26,n27,n28,n33,n36}
44	n23	n28 (D out)	dead path	n15	{n24,n25,n26,n27,n28,n33,n36}
45	n24	n37 (E in)	ok	n15	{n25,n26,n27,n28,n33,n36,n37}
46	n24	n28 (D out)	dead path	n15	{n25,n26,n27,n28,n33,n36,n37}
47	n25	n38 (E in)	overload	n15	{n26,n27,n28,n33,n36,n37}
48	n25	n28 (D out)	dead path	n15	{n26,n27,n28,n33,n36,n37}
49	n26	n39 (E in)	ok	n15	{n27,n28,n33,n36,n37,n39}
50	n26	n28 (D out)	dead path	n15	{n27,n28,n33,n36,n37,n39}
51	n27	n40 (E in)	ok	n15	{n28,n33,n36,n37,n39,n40}
52	n27	n28 (D out)	dead path	n15	{n28,n33,n36,n37,n39,n40}
53	n28	n41 (E in)	ok	n15	{n33,n36,n37,n39,n40,n41}
54	n28	n28 (D out)	dead path	n15	{n33,n36,n37,n39,n40,n41}
55	n33	-	-	n15	{n36,n37,n39,n40,n41}
56	n36	-	-	n15	{n37,n39,n40,n41}
57	n37	-	-	n15	{n39,n40,n41}
58	n39	-	-	n15	{n40,n41}
59	n40	-	-	n15	{n41}
60	n41	-	-	n15	{}

Solusi berakhir pada simpul yang memiliki total profit tertinggi yaitu n15 dengan nilai total profit 29700. Sehingga didapatkan hasil sebagai berikut.

Label	Nama	Weight	Profit	Density	Ambil
A	Salak	850.0	20000.0	~23.52	Ya
B	Buah Naga	400.0	6200.0	15.5	Ya
C	Markisa	600.0	5000.0	~8.33	Tidak
D	Srikaya	500.0	3500.0	7.0	Ya
E	Delima	800.0	5500.0	6.875	Tidak



Daftar Barang

Kapasistas Knapsack
1750

Total Profit Maksimum
29700

Nama

Berat

Keuntungan

Salak
Weight Profit
850 20000
Density
23.53

Buah Naga
Weight Profit
400 6200
Density
15.50

Markisa
Weight Profit
600 5000
Density
8.33

Srikaya
Weight Profit
500 3500
Density
7.00

Delima
Weight Profit
800 5500
Density
6.88

C. Perhitungan dataset 3

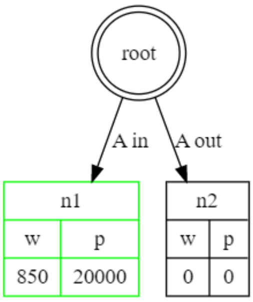
Kapasitas keranjang = 1600

Nama	Weight	Profit	Density
Salak	850.0	20000.0	~23.52
Delima	800.0	5500.0	6.875
Buah Naga	400.0	6200.0	15.5
Nanas	800.0	8000.0	10.0
Mangga	750.0	10000.0	~13.33
Blimbing	400.0	6200.0	15.5

Lakukan pengurutan berdasarkan *density* secara *descending* untuk mempermudah pencarian berdasarkan *greedy by density*.

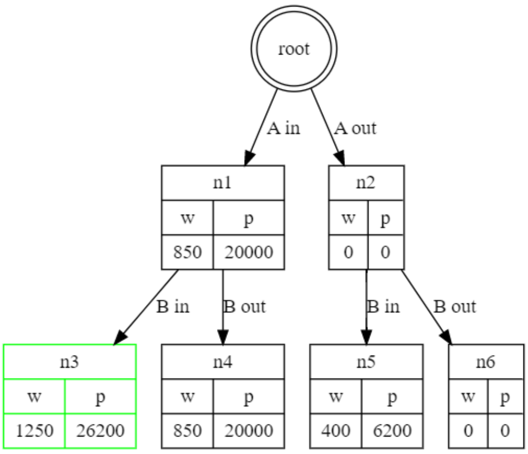
Label	Nama	Weight	Profit	Density
A	Salak	850.0	20000.0	~23.52
B	Buah Naga	400.0	6200.0	15.5
C	Blimbing	400.0	6200.0	15.5
D	Mangga	750.0	10000.0	~13.33
E	Nanas	800.0	8000.0	10.0
F	Delima	800.0	5500.0	6.875

Langkah pada *depth* 0



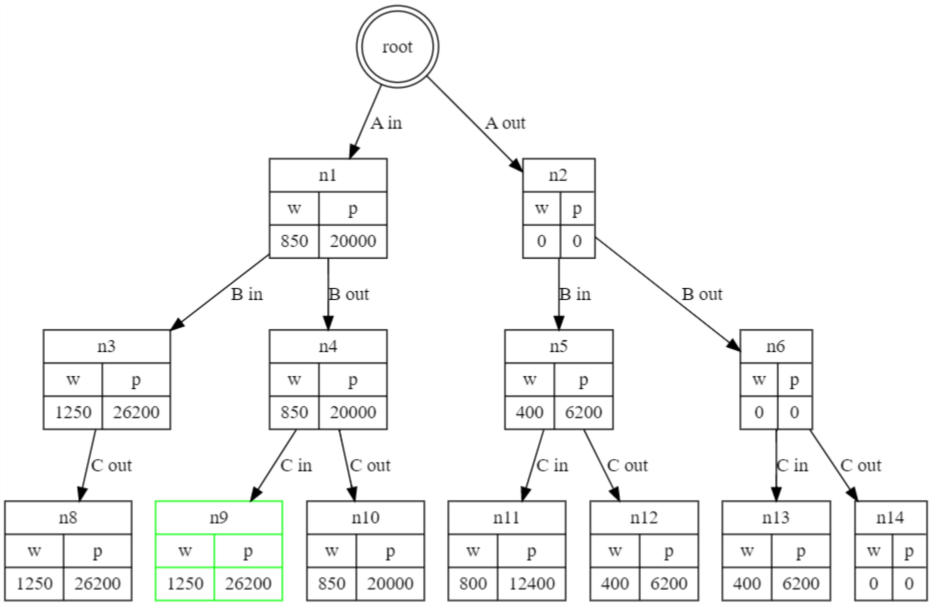
itr	current	branch	status	solution	queue
1	root	n1 (A in)	ok	n1	{n1}
2	root	n2 (A out)	ok	n1	{n1,n2}

Langkah pada *depth* 1



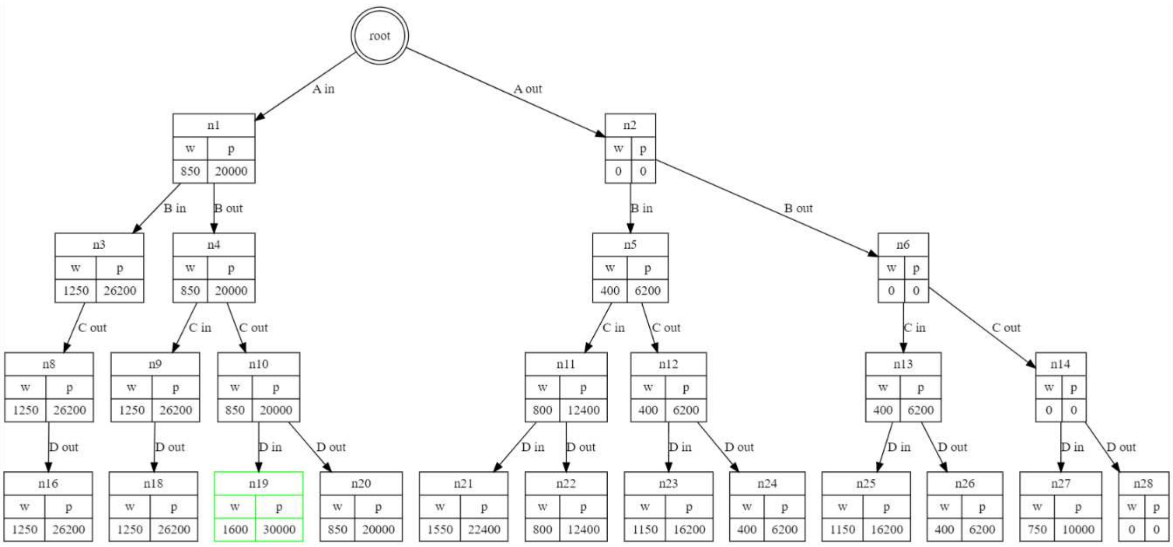
itr	current	branch	status	solution	queue
3	n1	n3 (B in)	ok	n3	{n2,n3}
4	n1	n4 (B out)	ok	n3	{n2,n3,n4}
5	n2	n5 (B in)	ok	n3	{n3,n4,n5}
6	n2	n6 (B out)	ok	n3	{n3,n4,n5,n6}

Langkah pada *depth* 2



itr	current	branch	status	solution	queue
7	n3	n7 (C in)	overload	n3	{n4,n5,n6}
8	n3	n8 (C out)	ok	n8	{n4,n5,n6,n8}
9	n4	n9 (C in)	ok	n9	{n5,n6,n8,n9}
10	n4	n10 (C out)	ok	n9	{n5,n6,n8,n9,n10}
11	n5	n11 (C in)	ok	n9	{n6,n8,n9,n10,n11}
12	n5	n12 (C out)	ok	n9	{n6,n8,n9,n10,n11,n12}
13	n6	n13 (C in)	ok	n9	{n8,n9,n10,n11,n12,n13}
14	n6	n14 (C out)	ok	n9	{n8,n9,n10,n11,n12,n13,n14}

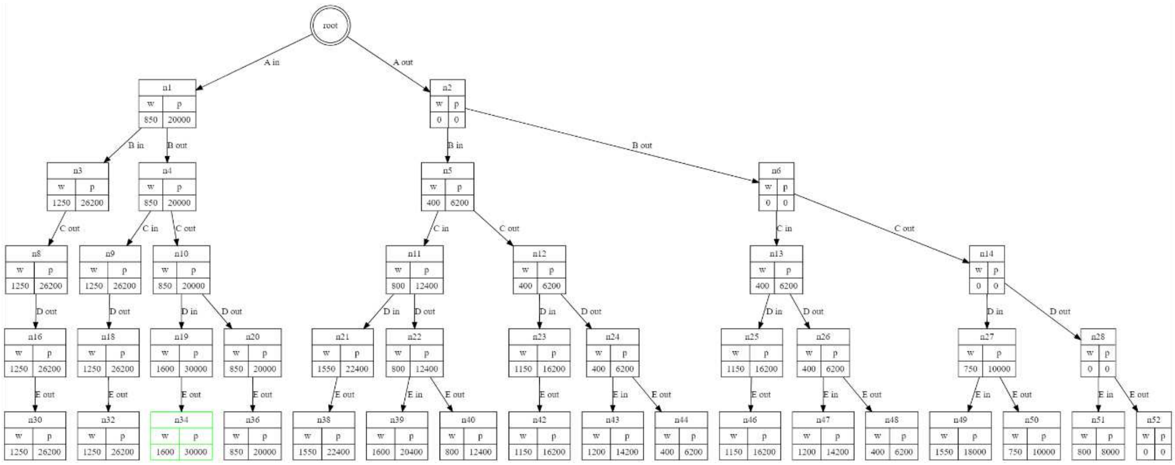
Langkah pada *depth* 3



itr	current	branch	status	solution	queue

15	n8	n15 (D in)	overload	n9	{n9,n10,n11,n12,n13,n14}
16	n8	n16 (D out)	ok	n16	{n9,n10,n11,n12,n13,n14,n16}
17	n9	n17 (D in)	overload	n16	{n10,n11,n12,n13,n14,n16}
18	n9	n18 (D out)	ok	n18	{n10,n11,n12,n13,n14,n16,n18}
19	n10	n19 (D in)	ok	n19	{n11,n12,n13,n14,n16,n18,n19}
20	n10	n20 (D out)	ok	n19	{n11,n12,n13,n14,n16,n18,n19,n20}
21	n11	n21 (D in)	ok	n19	{n12,n13,n14,n16,n18,n19,n20,n21}
22	n11	n22 (D out)	ok	n19	{n12,n13,n14,n16,n18,n19,n20,n21,n22}
23	n12	n23 (D in)	ok	n19	{n13,n14,n16,n18,n19,n20,n21,n22,n23}
24	n12	n24 (D out)	ok	n19	{n13,n14,n16,n18,n19,n20,n21,n22,n23,n24}
25	n13	n25 (D in)	ok	n19	{n14,n16,n18,n19,n20,n21,n22,n23,n24,n25}
26	n13	n26 (D out)	ok	n19	{n14,n16,n18,n19,n20,n21,n22,n23,n24,n25,n26}
27	n14	n27 (D in)	ok	n19	{n16,n18,n19,n20,n21,n22,n23,n24,n25,n26,n27}
28	n14	n28 (D out)	ok	n19	{n16,n18,n19,n20,n21,n22,n23,n24,n25,n26,n27,n28}

Langkah pada *depth* 4

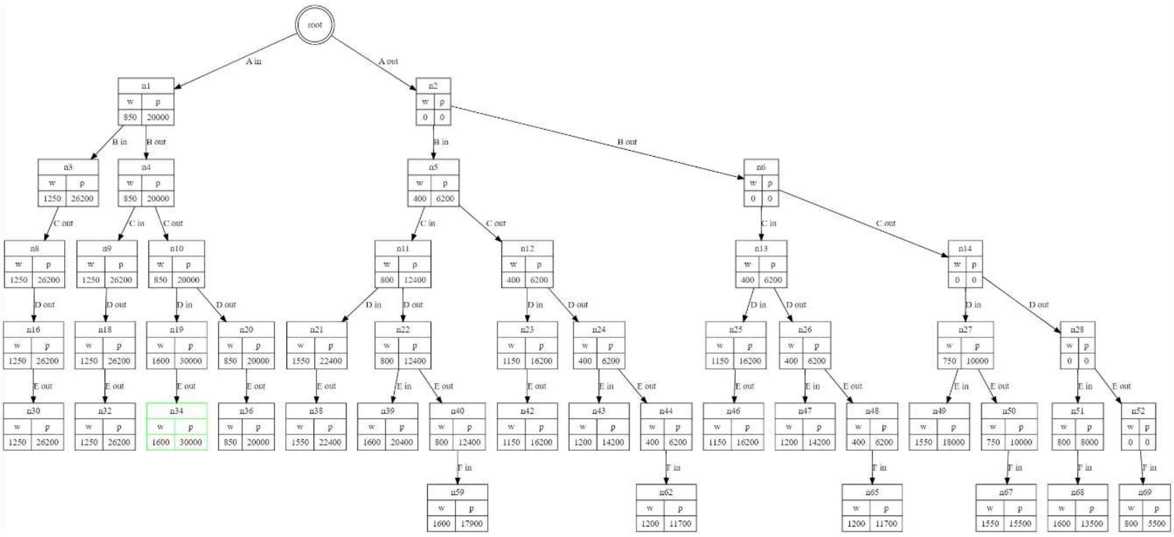


itr	current	branch	status	solution	queue

29	n16	n29 (E in)	overload	n19	{n18,n19,n20,n21,n22,n23,n24,n25,n26,n27,n28}
30	n16	n30 (E out)	ok	n19	{n18,n19,n20,n21,n22,n23,n24,n25,n26,n27,n28,n30}
31	n18	n31 (E in)	overload	n19	{n19,n20,n21,n22,n23,n24,n25,n26,n27,n28,n30}
32	n18	n32 (E out)	ok	n19	{n19,n20,n21,n22,n23,n24,n25,n26,n27,n28,n30,n32}
33	n19	n33 (E in)	overload	n19	{n20,n21,n22,n23,n24,n25,n26,n27,n28,n30,n32}
34	n19	n34 (E out)	ok	n34	{n20,n21,n22,n23,n24,n25,n26,n27,n28,n30,n32,n34}
35	n20	n35 (E in)	overload	n34	{n21,n22,n23,n24,n25,n26,n27,n28,n30,n32,n34}
36	n20	n36 (E out)	ok	n34	{n21,n22,n23,n24,n25,n26,n27,n28,n30,n32,n34,n36}
37	n21	n37 (E in)	overload	n34	{n22,n23,n24,n25,n26,n27,n28,n30,n32,n34,n36}
38	n21	n38 (E out)	ok	n34	{n22,n23,n24,n25,n26,n27,n28,n30,n32,n34,n36,n38}
39	n22	n39 (E in)	ok	n34	{n23,n24,n25,n26,n27,n28,n30,n32,n34,n36,n38,n39}
40	n22	n40 (E out)	ok	n34	{n23,n24,n25,n26,n27,n28,n30,n32,n34,n36,n38,n39,n40}
41	n23	n41 (E in)	overload	n34	{n24,n25,n26,n27,n28,n30,n32,n34,n36,n38,n39,n40}
42	n23	n42 (E out)	ok	n34	{n24,n25,n26,n27,n28,n30,n32,n34,n36,n38,n39,n40,n42}
43	n24	n43 (E in)	ok	n34	{n25,n26,n27,n28,n30,n32,n34,n36,n38,n39,n40,n42,n43}
44	n24	n44 (E out)	ok	n34	{n25,n26,n27,n28,n30,n32,n34,n36,n38,n39,n40,n42,n43,n44}

45	n25	n45 (E in)	overload	n34	{n26,n27,n28,n30,n32,n34,n36,n38,n39,n40,n42,n43,n44}
46	n25	n46 (E out)	ok	n34	{n26,n27,n28,n30,n32,n34,n36,n38,n39,n40,n42,n43,n44,n46}
47	n26	n47 (E in)	ok	n34	{n27,n28,n30,n32,n34,n36,n38,n39,n40,n42,n43,n44,n46,n47}
48	n26	n48 (E out)	ok	n34	{n27,n28,n30,n32,n34,n36,n38,n39,n40,n42,n43,n44,n46,n47,n48}
49	n27	n49 (E in)	ok	n34	{n28,n30,n32,n34,n36,n38,n39,n40,n42,n43,n44,n46,n47,n48,n49}
50	n27	n50 (E out)	ok	n34	{n28,n30,n32,n34,n36,n38,n39,n40,n42,n43,n44,n46,n47,n48,n49,n50}
51	n28	n51 (E in)	ok	n34	{n30,n32,n34,n36,n38,n39,n40,n42,n43,n44,n46,n47,n48,n49,n50,n51}
52	n28	n52 (E out)	ok	n34	{n30,n32,n34,n36,n38,n39,n40,n42,n43,n44,n46,n47,n48,n49,n50,n51,n52}

Langkah pada *depth* 5



itr	current	branch	status	solution	queue
53	n30	n53 (F in)	overload	n34	{n32,n34,n36,n38,n39,n40,n42,n43,n44,n46,n47,n48,n49,n50,n51,n52}
54	n30	n52 (E out)	dead path	n34	{n32,n34,n36,n38,n39,n40,n42,n43,n44,n46,n47,n48,n49,n50,n51,n52}

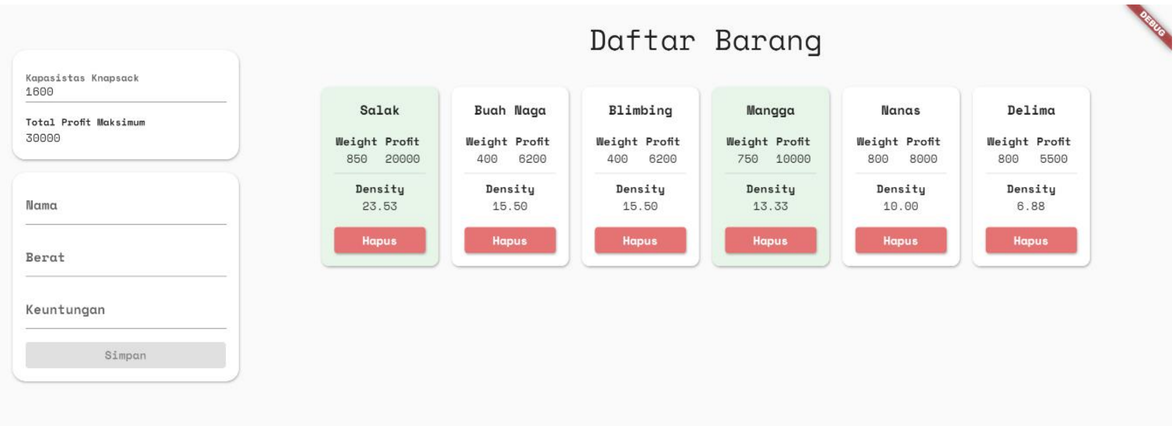
55	n32	n54 (F in)	overload	n34	{n34,n36,n38,n39,n40,n42,n43,n44,n46,n47,n48,n49,n50,n51,n52}
56	n32	n52 (E out)	dead path	n34	{n34,n36,n38,n39,n40,n42,n43,n44,n46,n47,n48,n49,n50,n51,n52}
57	n34	n55 (F in)	overload	n34	{n36,n38,n39,n40,n42,n43,n44,n46,n47,n48,n49,n50,n51,n52}
58	n34	n52 (E out)	dead path	n34	{n36,n38,n39,n40,n42,n43,n44,n46,n47,n48,n49,n50,n51,n52}
59	n36	n56 (F in)	overload	n34	{n38,n39,n40,n42,n43,n44,n46,n47,n48,n49,n50,n51,n52}
60	n36	n52 (E out)	dead path	n34	{n38,n39,n40,n42,n43,n44,n46,n47,n48,n49,n50,n51,n52}
61	n38	n57 (F in)	overload	n34	{n39,n40,n42,n43,n44,n46,n47,n48,n49,n50,n51,n52}
62	n38	n52 (E out)	dead path	n34	{n39,n40,n42,n43,n44,n46,n47,n48,n49,n50,n51,n52}
63	n39	n58 (F in)	overload	n34	{n40,n42,n43,n44,n46,n47,n48,n49,n50,n51,n52}
64	n39	n52 (E out)	dead path	n34	{n40,n42,n43,n44,n46,n47,n48,n49,n50,n51,n52}
65	n40	n59 (F in)	ok	n34	{n42,n43,n44,n46,n47,n48,n49,n50,n51,n52,n59}
66	n40	n52 (E out)	dead path	n34	{n42,n43,n44,n46,n47,n48,n49,n50,n51,n52,n59}
67	n42	n60 (F in)	overload	n34	{n43,n44,n46,n47,n48,n49,n50,n51,n52,n59}
68	n42	n52 (E out)	dead path	n34	{n43,n44,n46,n47,n48,n49,n50,n51,n52,n59}
69	n43	n61 (F in)	overload	n34	{n44,n46,n47,n48,n49,n50,n51,n52,n59}
70	n43	n52 (E out)	dead path	n34	{n44,n46,n47,n48,n49,n50,n51,n52,n59}

71	n44	n62 (F in)	ok	n34	{n46,n47,n48,n49,n50,n51,n52,n59,n62}
72	n44	n52 (E out)	dead path	n34	{n46,n47,n48,n49,n50,n51,n52,n59,n62}
73	n46	n63 (F in)	overload	n34	{n47,n48,n49,n50,n51,n52,n59,n62}
74	n46	n52 (E out)	dead path	n34	{n47,n48,n49,n50,n51,n52,n59,n62}
75	n47	n64 (F in)	overload	n34	{n48,n49,n50,n51,n52,n59,n62}
76	n47	n52 (E out)	dead path	n34	{n48,n49,n50,n51,n52,n59,n62}
77	n48	n65 (F in)	ok	n34	{n49,n50,n51,n52,n59,n62,n65}
78	n48	n52 (E out)	dead path	n34	{n49,n50,n51,n52,n59,n62,n65}
79	n49	n66 (F in)	overload	n34	{n50,n51,n52,n59,n62,n65}
80	n49	n52 (E out)	dead path	n34	{n50,n51,n52,n59,n62,n65}
81	n50	n67 (F in)	ok	n34	{n51,n52,n59,n62,n65,n67}
82	n50	n52 (E out)	dead path	n34	{n51,n52,n59,n62,n65,n67}
83	n51	n68 (F in)	ok	n34	{n52,n59,n62,n65,n67,n68}
84	n51	n52 (E out)	dead path	n34	{n52,n59,n62,n65,n67,n68}
85	n52	n69 (F in)	ok	n34	{n59,n62,n65,n67,n68,n69}
86	n52	n52 (E out)	dead path	n34	{n59,n62,n65,n67,n68,n69}
87	n59	-	-	n34	{n62,n65,n67,n68,n69}
88	n62	-	-	n34	{n65,n67,n68,n69}
89	n65	-	-	n34	{n67,n68,n69}
90	n67	-	-	n34	{n68,n69}
91	n68	-	-	n34	{n69}
92	n69	-	-	n34	{}

Solusi berakhir pada simpul yang memiliki total profit tertinggi yaitu n34 dengan nilai total profit 30000. Sehingga didapatkan hasil sebagai berikut.

Label	**Nama**	**Weight**	**Profit**	**Density**	**Ambil**
A	Salak	850.0	20000.0	~23.52	Ya
B	Buah Naga	400.0	6200.0	15.5	Tidak
C	Blimbing	400.0	6200.0	15.5	Tidak
D	Mangga	750.0	10000.0	~13.33	Ya
E	Nanas	800.0	8000.0	10.0	Tidak

F	Delima	800.0	5500.0	6.875	Tidak


Daftar Barang

Kapasistas Knapsack
1600

Total Profit Maksimum
30000

Nama

Berat

Keuntungan

Salak	Buah Naga	Blimbing	Mangga	Nanas	Delima
Weight Profit 850 20000	Weight Profit 400 6200	Weight Profit 400 6200	Weight Profit 750 10000	Weight Profit 800 8000	Weight Profit 800 5500
Density 23.53	Density 15.50	Density 15.50	Density 13.33	Density 10.00	Density 6.88

D. Perhitungan dataset 4

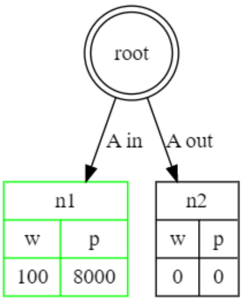
Kapasitas keranjang = 1400

Nama	**Weight**	**Profit**	**Density**
Buah Naga	400.0	6200.0	15.5
Anggur	100.0	8000.0	80.0
Pisang	700.0	12000.0	~17.14
Kurma	800.0	12000.0	15.0
Jeruk	250.0	3800.0	15.2
Kelengkeng	750.0	13300.0	~17.73

Lakukan pengurutan berdasarkan *density* secara *descending* untuk mempermudah pencarian berdasarkan *greedy by density*.

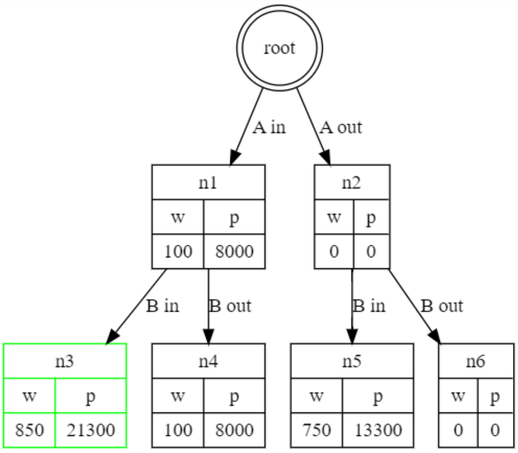
Label	**Nama**	**Weight**	**Profit**	**Density**
A	Anggur	100.0	8000.0	80.0
B	Kelengkeng	750.0	13300.0	~17.73
C	Pisang	700.0	12000.0	~17.14
D	Buah Naga	400.0	6200.0	15.5
E	Jeruk	250.0	3800.0	15.2
F	Kurma	800.0	12000.0	15.0

Langkah pada *depth* 0



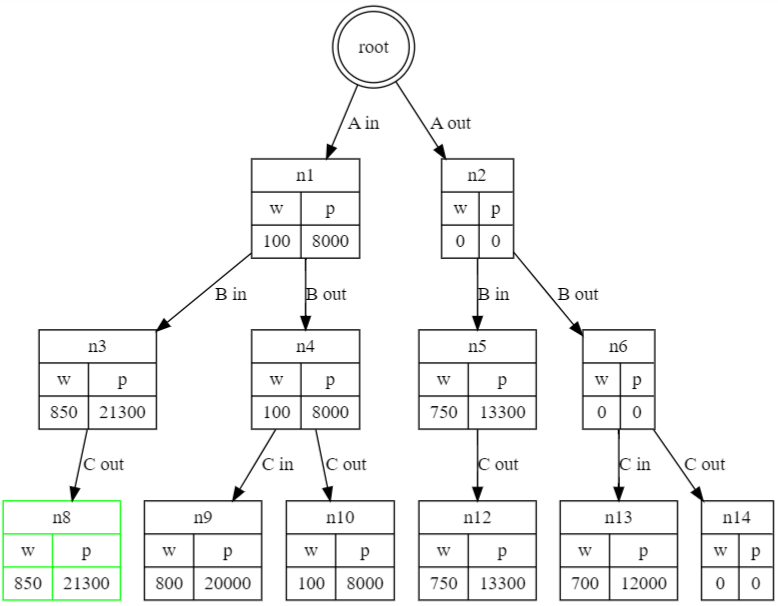
itr	current	branch	status	solution	queue
1	root	n1 (A in)	ok	n1	{n1}
2	root	n2 (A out)	ok	n1	{n1,n2}

Langkah pada *depth* 1



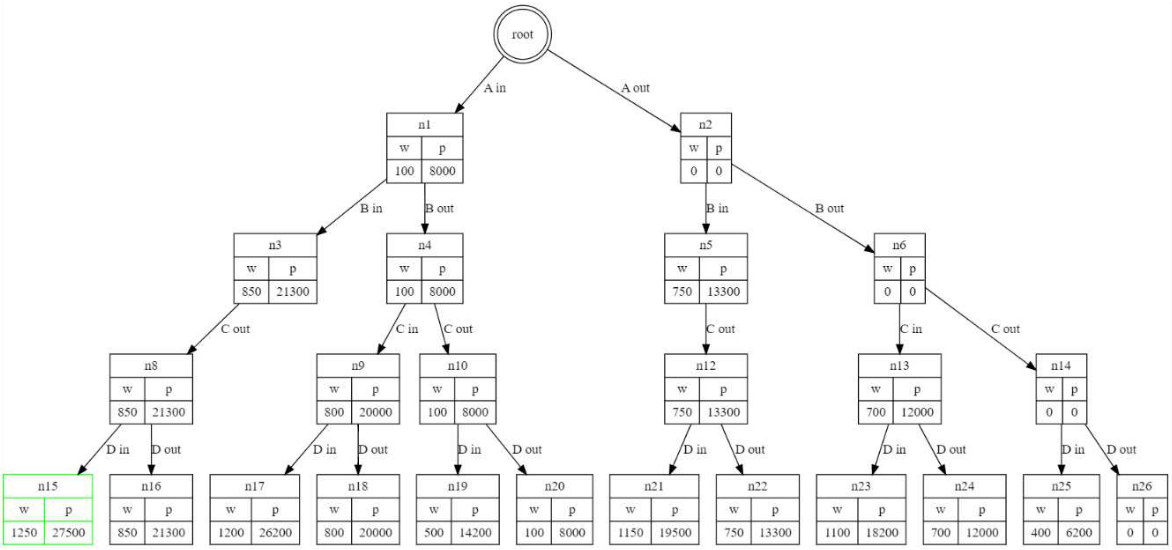
itr	current	branch	status	solution	queue
3	n1	n3 (B in)	ok	n3	{n2,n3}
4	n1	n4 (B out)	ok	n3	{n2,n3,n4}
5	n2	n5 (B in)	ok	n3	{n3,n4,n5}
6	n2	n6 (B out)	ok	n3	{n3,n4,n5,n6}

Langkah pada *depth* 2



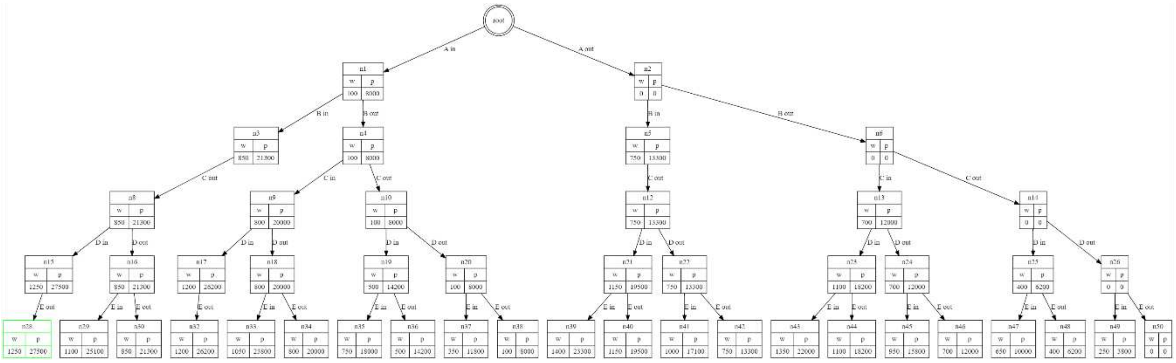
itr	current	branch	status	solution	queue
7	n3	n7 (C in)	overload	n3	{n4,n5,n6}
8	n3	n8 (C out)	ok	n8	{n4,n5,n6,n8}
9	n4	n9 (C in)	ok	n8	{n5,n6,n8,n9}
10	n4	n10 (C out)	ok	n8	{n5,n6,n8,n9,n10}
11	n5	n11 (C in)	overload	n8	{n6,n8,n9,n10}
12	n5	n12 (C out)	ok	n8	{n6,n8,n9,n10,n12}
13	n6	n13 (C in)	ok	n8	{n8,n9,n10,n12,n13}
14	n6	n14 (C out)	ok	n8	{n8,n9,n10,n12,n13,n14}

Langkah pada *depth* 3



itr	current	branch	status	solution	queue
15	n8	n15 (D in)	ok	n15	{n9,n10,n12,n13,n14,n15}
16	n8	n16 (D out)	ok	n15	{n9,n10,n12,n13,n14,n15,n16}
17	n9	n17 (D in)	ok	n15	{n10,n12,n13,n14,n15,n16,n17}
18	n9	n18 (D out)	ok	n15	{n10,n12,n13,n14,n15,n16,n17,n18}
19	n10	n19 (D in)	ok	n15	{n12,n13,n14,n15,n16,n17,n18,n19}
20	n10	n20 (D out)	ok	n15	{n12,n13,n14,n15,n16,n17,n18,n19,n20}
21	n12	n21 (D in)	ok	n15	{n13,n14,n15,n16,n17,n18,n19,n20,n21}
22	n12	n22 (D out)	ok	n15	{n13,n14,n15,n16,n17,n18,n19,n20,n21,n22}
23	n13	n23 (D in)	ok	n15	{n14,n15,n16,n17,n18,n19,n20,n21,n22,n23}
24	n13	n24 (D out)	ok	n15	{n14,n15,n16,n17,n18,n19,n20,n21,n22,n23,n24}
25	n14	n25 (D in)	ok	n15	{n15,n16,n17,n18,n19,n20,n21,n22,n23,n24,n25}
26	n14	n26 (D out)	ok	n15	{n15,n16,n17,n18,n19,n20,n21,n22,n23,n24,n25,n26}

Langkah pada *depth* 4

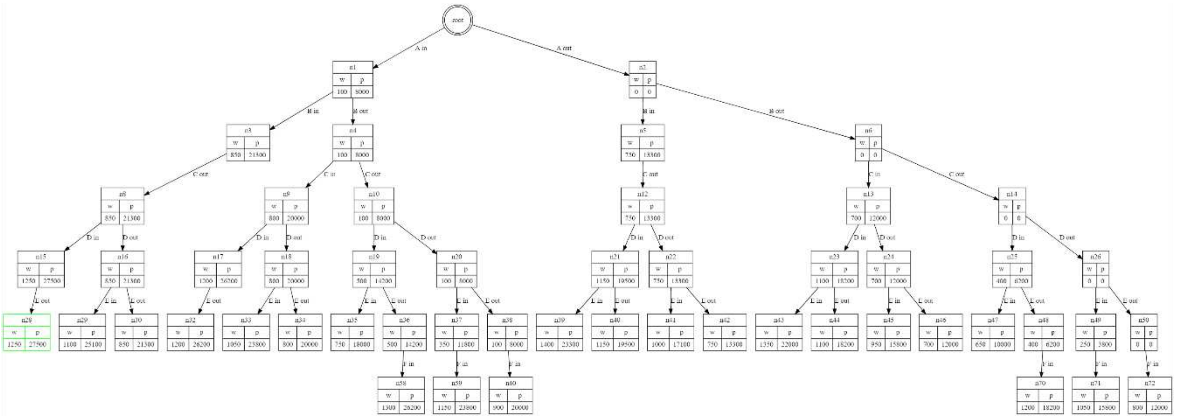


itr	current	branch	status	solution	queue
27	n15	n27 (E in)	overload	n15	{n16,n17,n18,n19,n20,n21,n22,n23,n24,n25,n26}

28	n15	n28 (E out)	ok	n28	{n16,n17,n18,n19,n20,n21,n22,n23,n24,n25,n26,n28}
29	n16	n29 (E in)	ok	n28	{n17,n18,n19,n20,n21,n22,n23,n24,n25,n26,n28,n29}
30	n16	n30 (E out)	ok	n28	{n17,n18,n19,n20,n21,n22,n23,n24,n25,n26,n28,n29,n30}
31	n17	n31 (E in)	overload	n28	{n18,n19,n20,n21,n22,n23,n24,n25,n26,n28,n29,n30}
32	n17	n32 (E out)	ok	n28	{n18,n19,n20,n21,n22,n23,n24,n25,n26,n28,n29,n30,n32}
33	n18	n33 (E in)	ok	n28	{n19,n20,n21,n22,n23,n24,n25,n26,n28,n29,n30,n32,n33}
34	n18	n34 (E out)	ok	n28	{n19,n20,n21,n22,n23,n24,n25,n26,n28,n29,n30,n32,n33,n34}
35	n19	n35 (E in)	ok	n28	{n20,n21,n22,n23,n24,n25,n26,n28,n29,n30,n32,n33,n34,n35}
36	n19	n36 (E out)	ok	n28	{n20,n21,n22,n23,n24,n25,n26,n28,n29,n30,n32,n33,n34,n35,n36}
37	n20	n37 (E in)	ok	n28	{n21,n22,n23,n24,n25,n26,n28,n29,n30,n32,n33,n34,n35,n36,n37}
38	n20	n38 (E out)	ok	n28	{n21,n22,n23,n24,n25,n26,n28,n29,n30,n32,n33,n34,n35,n36,n37,n38}
39	n21	n39 (E in)	ok	n28	{n22,n23,n24,n25,n26,n28,n29,n30,n32,n33,n34,n35,n36,n37,n38,n39}
40	n21	n40 (E out)	ok	n28	{n22,n23,n24,n25,n26,n28,n29,n30,n32,n33,n34,n35,n36,n37,n38,n39,n40}
41	n22	n41 (E in)	ok	n28	{n23,n24,n25,n26,n28,n29,n30,n32,n33,n34,n35,n36,n37,n38,n39,n40,n41}
42	n22	n42 (E out)	ok	n28	{n23,n24,n25,n26,n28,n29,n30,n32,n33,n34,n35,n36,n37,n38,n39,n40,n41,n42}

43	n23	n43 (E in)	ok	n28	{n24,n25,n26,n28,n29,n30,n32,n33,n34,n35,n36,n37,n38,n39,n40,n41,n42,n43}
44	n23	n44 (E out)	ok	n28	{n24,n25,n26,n28,n29,n30,n32,n33,n34,n35,n36,n37,n38,n39,n40,n41,n42,n43,n44}
45	n24	n45 (E in)	ok	n28	{n25,n26,n28,n29,n30,n32,n33,n34,n35,n36,n37,n38,n39,n40,n41,n42,n43,n44,n45}
46	n24	n46 (E out)	ok	n28	{n25,n26,n28,n29,n30,n32,n33,n34,n35,n36,n37,n38,n39,n40,n41,n42,n43,n44,n45,n46}
47	n25	n47 (E in)	ok	n28	{n26,n28,n29,n30,n32,n33,n34,n35,n36,n37,n38,n39,n40,n41,n42,n43,n44,n45,n46,n47}
48	n25	n48 (E out)	ok	n28	{n26,n28,n29,n30,n32,n33,n34,n35,n36,n37,n38,n39,n40,n41,n42,n43,n44,n45,n46,n47,n48}
49	n26	n49 (E in)	ok	n28	{n28,n29,n30,n32,n33,n34,n35,n36,n37,n38,n39,n40,n41,n42,n43,n44,n45,n46,n47,n48,n49}
50	n26	n50 (E out)	ok	n28	{n28,n29,n30,n32,n33,n34,n35,n36,n37,n38,n39,n40,n41,n42,n43,n44,n45,n46,n47,n48,n49,n50}

Langkah pada *depth* 5



itr	current	branch	status	solution	queue
51	n28	n51 (F in)	overload	n28	{n29,n30,n32,n33,n34,n35,n36,n37,n38,n39,n40,n41,n42,n43,n44,n45,n46,n47,n48,n49,n50}
52	n28	n50 (E out)	dead path	n28	{n29,n30,n32,n33,n34,n35,n36,n37,n38,n39,n40,n41,n42,n43,n44,n45,n46,n47,n48,n49,n50}
53	n29	n52 (F in)	overload	n28	{n30,n32,n33,n34,n35,n36,n37,n38,n39,n40,n41,n42,n43,n44,n45,n46,n47,n48,n49,n50}
54	n29	n50 (E out)	dead path	n28	{n30,n32,n33,n34,n35,n36,n37,n38,n39,n40,n41,n42,n43,n44,n45,n46,n47,n48,n49,n50}
55	n30	n53 (F in)	overload	n28	{n32,n33,n34,n35,n36,n37,n38,n39,n40,n41,n42,n43,n44,n45,n46,n47,n48,n49,n50}
56	n30	n50 (E out)	dead path	n28	{n32,n33,n34,n35,n36,n37,n38,n39,n40,n41,n42,n43,n44,n45,n46,n47,n48,n49,n50}
57	n32	n54 (F in)	overload	n28	{n33,n34,n35,n36,n37,n38,n39,n40,n41,n42,n43,n44,n45,n46,n47,n48,n49,n50}
58	n32	n50 (E out)	dead path	n28	{n33,n34,n35,n36,n37,n38,n39,n40,n41,n42,n43,n44,n45,n46,n47,n48,n49,n50}
59	n33	n55 (F in)	overload	n28	{n34,n35,n36,n37,n38,n39,n40,n41,n42,n43,n44,n45,n46,n47,n48,n49,n50}
60	n33	n50 (E out)	dead path	n28	{n34,n35,n36,n37,n38,n39,n40,n41,n42,n43,n44,n45,n46,n47,n48,n49,n50}
61	n34	n56 (F in)	overload	n28	{n35,n36,n37,n38,n39,n40,n41,n42,n43,n44,n45,n46,n47,n48,n49,n50}
62	n34	n50 (E out)	dead path	n28	{n35,n36,n37,n38,n39,n40,n41,n42,n43,n44,n45,n46,n47,n48,n49,n50}

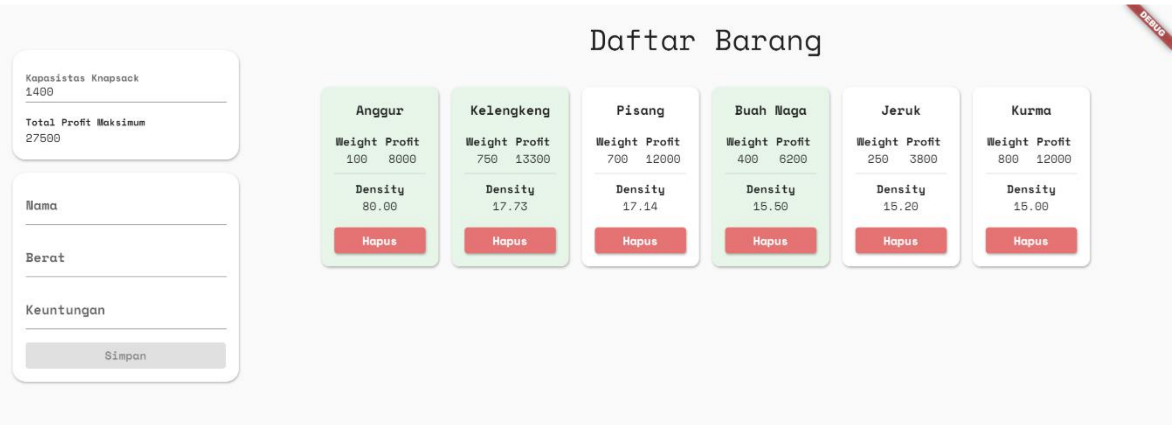
63	n35	n57 (F in)	overload	n28	{n36,n37,n38,n39,n40,n41,n42,n43,n44,n45,n46,n47,n48,n49,n50}
64	n35	n50 (E out)	dead path	n28	{n36,n37,n38,n39,n40,n41,n42,n43,n44,n45,n46,n47,n48,n49,n50}
65	n36	n58 (F in)	ok	n28	{n37,n38,n39,n40,n41,n42,n43,n44,n45,n46,n47,n48,n49,n50,n58}
66	n36	n50 (E out)	dead path	n28	{n37,n38,n39,n40,n41,n42,n43,n44,n45,n46,n47,n48,n49,n50,n58}
67	n37	n59 (F in)	ok	n28	{n38,n39,n40,n41,n42,n43,n44,n45,n46,n47,n48,n49,n50,n58,n59}
68	n37	n50 (E out)	dead path	n28	{n38,n39,n40,n41,n42,n43,n44,n45,n46,n47,n48,n49,n50,n58,n59}
69	n38	n60 (F in)	ok	n28	{n39,n40,n41,n42,n43,n44,n45,n46,n47,n48,n49,n50,n58,n59,n60}
70	n38	n50 (E out)	dead path	n28	{n39,n40,n41,n42,n43,n44,n45,n46,n47,n48,n49,n50,n58,n59,n60}
71	n39	n61 (F in)	overload	n28	{n40,n41,n42,n43,n44,n45,n46,n47,n48,n49,n50,n58,n59,n60}
72	n39	n50 (E out)	dead path	n28	{n40,n41,n42,n43,n44,n45,n46,n47,n48,n49,n50,n58,n59,n60}
73	n40	n62 (F in)	overload	n28	{n41,n42,n43,n44,n45,n46,n47,n48,n49,n50,n58,n59,n60}
74	n40	n50 (E out)	dead path	n28	{n41,n42,n43,n44,n45,n46,n47,n48,n49,n50,n58,n59,n60}
75	n41	n63 (F in)	overload	n28	{n42,n43,n44,n45,n46,n47,n48,n49,n50,n58,n59,n60}
76	n41	n50 (E out)	dead path	n28	{n42,n43,n44,n45,n46,n47,n48,n49,n50,n58,n59,n60}
77	n42	n64 (F in)	overload	n28	{n43,n44,n45,n46,n47,n48,n49,n50,n58,n59,n60}
78	n42	n50 (E out)	dead path	n28	{n43,n44,n45,n46,n47,n48,n49,n50,n58,n59,n60}

79	n43	n65 (F in)	overload	n28	{n44,n45,n46,n47,n48,n49,n50,n58,n59,n60}
80	n43	n50 (E out)	dead path	n28	{n44,n45,n46,n47,n48,n49,n50,n58,n59,n60}
81	n44	n66 (F in)	overload	n28	{n45,n46,n47,n48,n49,n50,n58,n59,n60}
82	n44	n50 (E out)	dead path	n28	{n45,n46,n47,n48,n49,n50,n58,n59,n60}
83	n45	n67 (F in)	overload	n28	{n46,n47,n48,n49,n50,n58,n59,n60}
84	n45	n50 (E out)	dead path	n28	{n46,n47,n48,n49,n50,n58,n59,n60}
85	n46	n68 (F in)	overload	n28	{n47,n48,n49,n50,n58,n59,n60}
86	n46	n50 (E out)	dead path	n28	{n47,n48,n49,n50,n58,n59,n60}
87	n47	n69 (F in)	overload	n28	{n48,n49,n50,n58,n59,n60}
88	n47	n50 (E out)	dead path	n28	{n48,n49,n50,n58,n59,n60}
89	n48	n70 (F in)	ok	n28	{n49,n50,n58,n59,n60,n70}
90	n48	n50 (E out)	dead path	n28	{n49,n50,n58,n59,n60,n70}
91	n49	n71 (F in)	ok	n28	{n50,n58,n59,n60,n70,n71}
92	n49	n50 (E out)	dead path	n28	{n50,n58,n59,n60,n70,n71}
93	n50	n72 (F in)	ok	n28	{n58,n59,n60,n70,n71,n72}
94	n50	n50 (E out)	dead path	n28	{n58,n59,n60,n70,n71,n72}
95	n58	-	-	n28	{n59,n60,n70,n71,n72}
96	n59	-	-	n28	{n60,n70,n71,n72}
97	n60	-	-	n28	{n70,n71,n72}
98	n70	-	-	n28	{n71,n72}
99	n71	-	-	n28	{n72}
100	n72	-	-	n28	{}

Solusi berakhir pada simpul yang memiliki total profit tertinggi yaitu n28 dengan nilai total profit 27500. Sehingga didapatkan hasil sebagai berikut.

Label	Nama	Weight	Profit	Density	Ambil
A	Anggur	100.0	8000.0	80.0	Ya
B	Kelengkeng	750.0	13300.0	~17.73	Ya
C	Pisang	700.0	12000.0	~17.14	Tidak

D	Buah Naga	400.0	6200.0	15.5	Ya
E	Jeruk	250.0	3800.0	15.2	Tidak
F	Kurma	800.0	12000.0	15.0	Tidak



Daftar Barang

Kapasistas Knapsack
1400

Total Profit Maksimum
27500

Nama

Berat

Keuntungan

Simpan

Anggur
Weight Profit
100 8000
Density
80.00
Hapus

Kelengkeng
Weight Profit
750 13300
Density
17.73
Hapus

Pisang
Weight Profit
700 12000
Density
17.14
Hapus

Buah Naga
Weight Profit
400 6200
Density
15.50
Hapus

Jeruk
Weight Profit
250 3800
Density
15.20
Hapus

Kurma
Weight Profit
800 12000
Density
15.00
Hapus

E. Perhitungan dataset 5

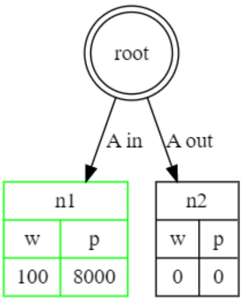
Kapasitas keranjang = 2000

Nama	Weight	Profit	Density
Blimbing	400.0	6200.0	15.5
Strawberry	500.0	14500.0	29.0
Jambu	600.0	11500.0	~19.16
Anggur	100.0	8000.0	80.0
Salak	850.0	20000.0	~23.52
Nanas	800.0	8000.0	10.0

Lakukan pengurutan berdasarkan *density* secara *descending* untuk mempermudah pencarian berdasarkan *greedy by density*.

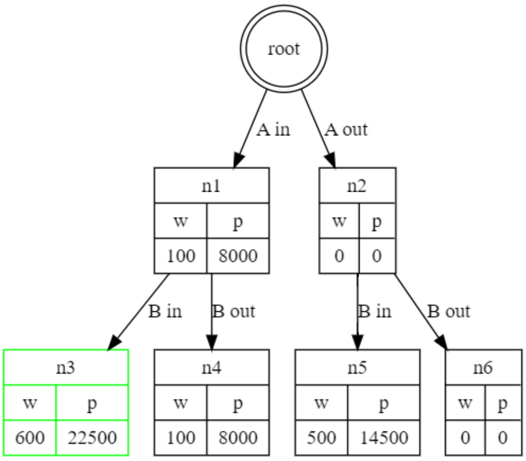
Label	Nama	Weight	Profit	Density
A	Anggur	100.0	8000.0	80.0
B	Strawberry	500.0	14500.0	29.0
C	Salak	850.0	20000.0	~23.52
D	Jambu	600.0	11500.0	~19.16
E	Blimbing	400.0	6200.0	15.5
F	Nanas	800.0	8000.0	10.0

Langkah pada *depth* 0



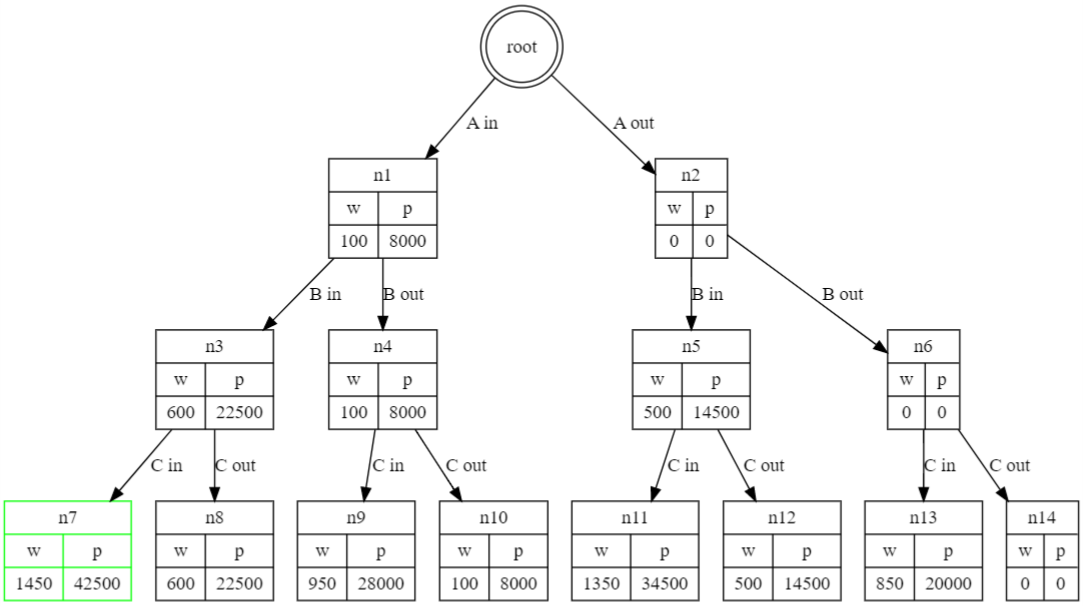
itr	current	branch	status	solution	queue
1	root	n1 (A in)	ok	n1	{n1}
2	root	n2 (A out)	ok	n1	{n1,n2}

Langkah pada *depth* 1



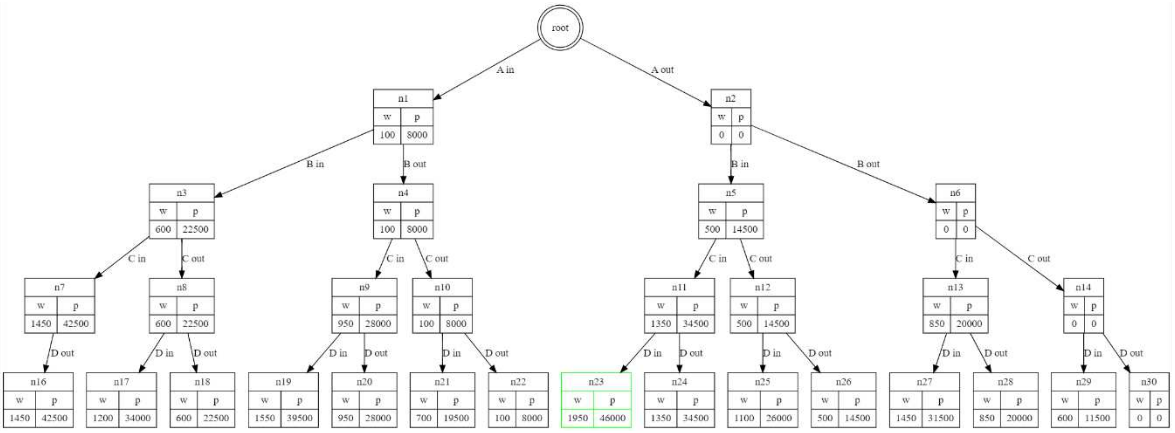
itr	current	branch	status	solution	queue
3	n1	n3 (B in)	ok	n3	{n2,n3}
4	n1	n4 (B out)	ok	n3	{n2,n3,n4}
5	n2	n5 (B in)	ok	n3	{n3,n4,n5}
6	n2	n6 (B out)	ok	n3	{n3,n4,n5,n6}

Langkah pada *depth* 2



itr	current	branch	status	solution	queue
7	n3	n7 (C in)	ok	n7	{n4,n5,n6,n7}
8	n3	n8 (C out)	ok	n7	{n4,n5,n6,n7,n8}
9	n4	n9 (C in)	ok	n7	{n5,n6,n7,n8,n9}
10	n4	n10 (C out)	ok	n7	{n5,n6,n7,n8,n9,n10}
11	n5	n11 (C in)	ok	n7	{n6,n7,n8,n9,n10,n11}
12	n5	n12 (C out)	ok	n7	{n6,n7,n8,n9,n10,n11,n12}
13	n6	n13 (C in)	ok	n7	{n7,n8,n9,n10,n11,n12,n13}
14	n6	n14 (C out)	ok	n7	{n7,n8,n9,n10,n11,n12,n13,n14}

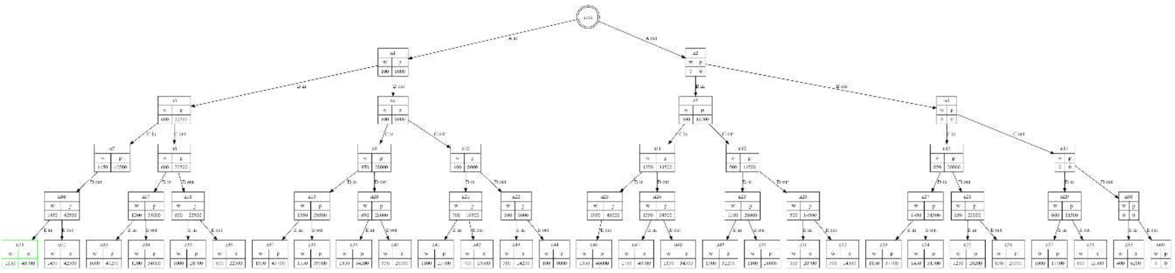
Langkah pada *depth* 3



itr	current	branch	status	solution	queue
15	n7	n15 (D in)	overload	n7	{n8,n9,n10,n11,n12,n13,n14}
16	n7	n16 (D out)	ok	n16	{n8,n9,n10,n11,n12,n13,n14,n16}

17	n8	n17 (D in)	ok	n16	{n9,n10,n11,n12,n13,n14,n16,n17}
18	n8	n18 (D out)	ok	n16	{n9,n10,n11,n12,n13,n14,n16,n17,n18}
19	n9	n19 (D in)	ok	n16	{n10,n11,n12,n13,n14,n16,n17,n18,n19}
20	n9	n20 (D out)	ok	n16	{n10,n11,n12,n13,n14,n16,n17,n18,n19,n20}
21	n10	n21 (D in)	ok	n16	{n11,n12,n13,n14,n16,n17,n18,n19,n20,n21}
22	n10	n22 (D out)	ok	n16	{n11,n12,n13,n14,n16,n17,n18,n19,n20,n21,n22}
23	n11	n23 (D in)	ok	n23	{n12,n13,n14,n16,n17,n18,n19,n20,n21,n22,n23}
24	n11	n24 (D out)	ok	n23	{n12,n13,n14,n16,n17,n18,n19,n20,n21,n22,n23,n24}
25	n12	n25 (D in)	ok	n23	{n13,n14,n16,n17,n18,n19,n20,n21,n22,n23,n24,n25}
26	n12	n26 (D out)	ok	n23	{n13,n14,n16,n17,n18,n19,n20,n21,n22,n23,n24,n25,n26}
27	n13	n27 (D in)	ok	n23	{n14,n16,n17,n18,n19,n20,n21,n22,n23,n24,n25,n26,n27}
28	n13	n28 (D out)	ok	n23	{n14,n16,n17,n18,n19,n20,n21,n22,n23,n24,n25,n26,n27,n28}
29	n14	n29 (D in)	ok	n23	{n16,n17,n18,n19,n20,n21,n22,n23,n24,n25,n26,n27,n28,n29}
30	n14	n30 (D out)	ok	n23	{n16,n17,n18,n19,n20,n21,n22,n23,n24,n25,n26,n27,n28,n29,n30}

Langkah pada *depth* 4

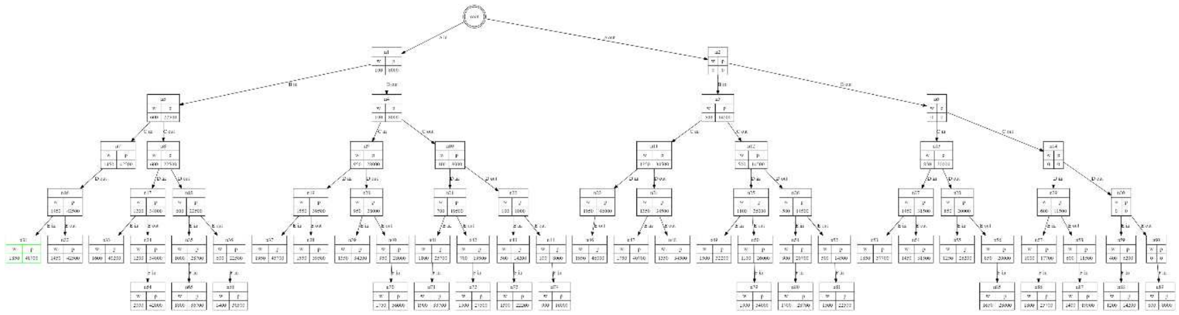


itr	current	branch	status	solution	queue
31	n16	n31 (E in)	ok	n31	{n17,n18,n19,n20,n21,n22,n23,n24,n25,n26,n27,n28,n29,n30,n31}
32	n16	n32 (E out)	ok	n31	{n17,n18,n19,n20,n21,n22,n23,n24,n25,n26,n27,n28,n29,n30,n31,n32}
33	n17	n33 (E in)	ok	n31	{n18,n19,n20,n21,n22,n23,n24,n25,n26,n27,n28,n29,n30,n31,n32,n33}
34	n17	n34 (E out)	ok	n31	{n18,n19,n20,n21,n22,n23,n24,n25,n26,n27,n28,n29,n30,n31,n32,n33,n34}
35	n18	n35 (E in)	ok	n31	{n19,n20,n21,n22,n23,n24,n25,n26,n27,n28,n29,n30,n31,n32,n33,n34,n35}
36	n18	n36 (E out)	ok	n31	{n19,n20,n21,n22,n23,n24,n25,n26,n27,n28,n29,n30,n31,n32,n33,n34,n35,n36}
37	n19	n37 (E in)	ok	n31	{n20,n21,n22,n23,n24,n25,n26,n27,n28,n29,n30,n31,n32,n33,n34,n35,n36,n37}
38	n19	n38 (E out)	ok	n31	{n20,n21,n22,n23,n24,n25,n26,n27,n28,n29,n30,n31,n32,n33,n34,n35,n36,n37,n38}
39	n20	n39 (E in)	ok	n31	{n21,n22,n23,n24,n25,n26,n27,n28,n29,n30,n31,n32,n33,n34,n35,n36,n37,n38,n39}
40	n20	n40 (E out)	ok	n31	{n21,n22,n23,n24,n25,n26,n27,n28,n29,n30,n31,n32,n33,n34,n35,n36,n37,n38,n39,n40}

41	n21	n41 (E in)	ok	n31	{n22,n23,n24,n25,n26,n27,n28,n29,n30,n31,n32,n33,n34,n35,n36,n37,n38,n39,n40,n41}
42	n21	n42 (E out)	ok	n31	{n22,n23,n24,n25,n26,n27,n28,n29,n30,n31,n32,n33,n34,n35,n36,n37,n38,n39,n40,n41,n42}
43	n22	n43 (E in)	ok	n31	{n23,n24,n25,n26,n27,n28,n29,n30,n31,n32,n33,n34,n35,n36,n37,n38,n39,n40,n41,n42,n43}
44	n22	n44 (E out)	ok	n31	{n23,n24,n25,n26,n27,n28,n29,n30,n31,n32,n33,n34,n35,n36,n37,n38,n39,n40,n41,n42,n43,n44}
45	n23	n45 (E in)	overload	n31	{n24,n25,n26,n27,n28,n29,n30,n31,n32,n33,n34,n35,n36,n37,n38,n39,n40,n41,n42,n43,n44}
46	n23	n46 (E out)	ok	n31	{n24,n25,n26,n27,n28,n29,n30,n31,n32,n33,n34,n35,n36,n37,n38,n39,n40,n41,n42,n43,n44,n46}
47	n24	n47 (E in)	ok	n31	{n25,n26,n27,n28,n29,n30,n31,n32,n33,n34,n35,n36,n37,n38,n39,n40,n41,n42,n43,n44,n46,n47}
48	n24	n48 (E out)	ok	n31	{n25,n26,n27,n28,n29,n30,n31,n32,n33,n34,n35,n36,n37,n38,n39,n40,n41,n42,n43,n44,n46,n47,n48}
49	n25	n49 (E in)	ok	n31	{n26,n27,n28,n29,n30,n31,n32,n33,n34,n35,n36,n37,n38,n39,n40,n41,n42,n43,n44,n46,n47,n48,n49}
50	n25	n50 (E out)	ok	n31	{n26,n27,n28,n29,n30,n31,n32,n33,n34,n35,n36,n37,n38,n39,n40,n41,n42,n43,n44,n46,n47,n48,n49,n50}
51	n26	n51 (E in)	ok	n31	{n27,n28,n29,n30,n31,n32,n33,n34,n35,n36,n37,n38,n39,n40,n41,n42,n43,n44,n46,n47,n48,n49,n50,n51}

52	n26	n52 (E out)	ok	n31	{n27,n28,n29,n30,n31,n32,n33,n34,n35,n36,n37,n38,n39,n40,n41,n42,n43,n44,n46,n47,n48,n49,n50,n51,n52}
53	n27	n53 (E in)	ok	n31	{n28,n29,n30,n31,n32,n33,n34,n35,n36,n37,n38,n39,n40,n41,n42,n43,n44,n46,n47,n48,n49,n50,n51,n52,n53}
54	n27	n54 (E out)	ok	n31	{n28,n29,n30,n31,n32,n33,n34,n35,n36,n37,n38,n39,n40,n41,n42,n43,n44,n46,n47,n48,n49,n50,n51,n52,n53,n54}
55	n28	n55 (E in)	ok	n31	{n29,n30,n31,n32,n33,n34,n35,n36,n37,n38,n39,n40,n41,n42,n43,n44,n46,n47,n48,n49,n50,n51,n52,n53,n54,n55}
56	n28	n56 (E out)	ok	n31	{n29,n30,n31,n32,n33,n34,n35,n36,n37,n38,n39,n40,n41,n42,n43,n44,n46,n47,n48,n49,n50,n51,n52,n53,n54,n55,n56}
57	n29	n57 (E in)	ok	n31	{n30,n31,n32,n33,n34,n35,n36,n37,n38,n39,n40,n41,n42,n43,n44,n46,n47,n48,n49,n50,n51,n52,n53,n54,n55,n56,n57}
58	n29	n58 (E out)	ok	n31	{n30,n31,n32,n33,n34,n35,n36,n37,n38,n39,n40,n41,n42,n43,n44,n46,n47,n48,n49,n50,n51,n52,n53,n54,n55,n56,n57,n58}
59	n30	n59 (E in)	ok	n31	{n31,n32,n33,n34,n35,n36,n37,n38,n39,n40,n41,n42,n43,n44,n46,n47,n48,n49,n50,n51,n52,n53,n54,n55,n56,n57,n58,n59}
60	n30	n60 (E out)	ok	n31	{n31,n32,n33,n34,n35,n36,n37,n38,n39,n40,n41,n42,n43,n44,n46,n47,n48,n49,n50,n51,n52,n53,n54,n55,n56,n57,n58,n59,n60}

Langkah pada *depth* 5



itr	current	branch	status	solution	queue
61	n31	n61 (F in)	overload	n31	{n32,n33,n34,n35,n36,n37,n38,n39,n40,n41,n42,n43,n44,n46,n47,n48,n49,n50,n51,n52,n53,n54,n55,n56,n57,n58,n59,n60}
62	n31	n60 (E out)	dead path	n31	{n32,n33,n34,n35,n36,n37,n38,n39,n40,n41,n42,n43,n44,n46,n47,n48,n49,n50,n51,n52,n53,n54,n55,n56,n57,n58,n59,n60}
63	n32	n62 (F in)	overload	n31	{n33,n34,n35,n36,n37,n38,n39,n40,n41,n42,n43,n44,n46,n47,n48,n49,n50,n51,n52,n53,n54,n55,n56,n57,n58,n59,n60}
64	n32	n60 (E out)	dead path	n31	{n33,n34,n35,n36,n37,n38,n39,n40,n41,n42,n43,n44,n46,n47,n48,n49,n50,n51,n52,n53,n54,n55,n56,n57,n58,n59,n60}
65	n33	n63 (F in)	overload	n31	{n34,n35,n36,n37,n38,n39,n40,n41,n42,n43,n44,n46,n47,n48,n49,n50,n51,n52,n53,n54,n55,n56,n57,n58,n59,n60}
66	n33	n60 (E out)	dead path	n31	{n34,n35,n36,n37,n38,n39,n40,n41,n42,n43,n44,n46,n47,n48,n49,n50,n51,n52,n53,n54,n55,n56,n57,n58,n59,n60}
67	n34	n64 (F in)	ok	n31	{n35,n36,n37,n38,n39,n40,n41,n42,n43,n44,n46,n47,n48,n49,n50,n51,n52,n53,n54,n55,n56,n57,n58,n59,n60,n64}

68	n34	n60 (E out)	dead path	n31	{n35,n36,n37,n38,n39,n40,n41,n42,n43,n44,n46,n47,n48,n49,n50,n51,n52,n53,n54,n55,n56,n57,n58,n59,n60,n64}
69	n35	n65 (F in)	ok	n31	{n36,n37,n38,n39,n40,n41,n42,n43,n44,n46,n47,n48,n49,n50,n51,n52,n53,n54,n55,n56,n57,n58,n59,n60,n64,n65}
70	n35	n60 (E out)	dead path	n31	{n36,n37,n38,n39,n40,n41,n42,n43,n44,n46,n47,n48,n49,n50,n51,n52,n53,n54,n55,n56,n57,n58,n59,n60,n64,n65}
71	n36	n66 (F in)	ok	n31	{n37,n38,n39,n40,n41,n42,n43,n44,n46,n47,n48,n49,n50,n51,n52,n53,n54,n55,n56,n57,n58,n59,n60,n64,n65,n66}
72	n36	n60 (E out)	dead path	n31	{n37,n38,n39,n40,n41,n42,n43,n44,n46,n47,n48,n49,n50,n51,n52,n53,n54,n55,n56,n57,n58,n59,n60,n64,n65,n66}
73	n37	n67 (F in)	overload	n31	{n38,n39,n40,n41,n42,n43,n44,n46,n47,n48,n49,n50,n51,n52,n53,n54,n55,n56,n57,n58,n59,n60,n64,n65,n66}
74	n37	n60 (E out)	dead path	n31	{n38,n39,n40,n41,n42,n43,n44,n46,n47,n48,n49,n50,n51,n52,n53,n54,n55,n56,n57,n58,n59,n60,n64,n65,n66}
75	n38	n68 (F in)	overload	n31	{n39,n40,n41,n42,n43,n44,n46,n47,n48,n49,n50,n51,n52,n53,n54,n55,n56,n57,n58,n59,n60,n64,n65,n66}
76	n38	n60 (E out)	dead path	n31	{n39,n40,n41,n42,n43,n44,n46,n47,n48,n49,n50,n51,n52,n53,n54,n55,n56,n57,n58,n59,n60,n64,n65,n66}
77	n39	n69 (F in)	overload	n31	{n40,n41,n42,n43,n44,n46,n47,n48,n49,n50,n51,n52,n53,n54,n55,n56,n57,n58,n59,n60,n64,n65,n66}
78	n39	n60 (E out)	dead path	n31	{n40,n41,n42,n43,n44,n46,n47,n48,n49,n50,n51,n52,n53,n54,n55,n56,n57,n58,n59,n60,n64,n65,n66}

79	n40	n70 (F in)	ok	n31	{n41,n42,n43,n44,n46,n47,n48,n49,n50,n51,n52,n53,n54,n55,n56,n57,n58,n59,n60,n64,n65,n66,n70}
80	n40	n60 (E out)	dead path	n31	{n41,n42,n43,n44,n46,n47,n48,n49,n50,n51,n52,n53,n54,n55,n56,n57,n58,n59,n60,n64,n65,n66,n70}
81	n41	n71 (F in)	ok	n31	{n42,n43,n44,n46,n47,n48,n49,n50,n51,n52,n53,n54,n55,n56,n57,n58,n59,n60,n64,n65,n66,n70,n71}
82	n41	n60 (E out)	dead path	n31	{n42,n43,n44,n46,n47,n48,n49,n50,n51,n52,n53,n54,n55,n56,n57,n58,n59,n60,n64,n65,n66,n70,n71}
83	n42	n72 (F in)	ok	n31	{n43,n44,n46,n47,n48,n49,n50,n51,n52,n53,n54,n55,n56,n57,n58,n59,n60,n64,n65,n66,n70,n71,n72}
84	n42	n60 (E out)	dead path	n31	{n43,n44,n46,n47,n48,n49,n50,n51,n52,n53,n54,n55,n56,n57,n58,n59,n60,n64,n65,n66,n70,n71,n72}
85	n43	n73 (F in)	ok	n31	{n44,n46,n47,n48,n49,n50,n51,n52,n53,n54,n55,n56,n57,n58,n59,n60,n64,n65,n66,n70,n71,n72,n73}
86	n43	n60 (E out)	dead path	n31	{n44,n46,n47,n48,n49,n50,n51,n52,n53,n54,n55,n56,n57,n58,n59,n60,n64,n65,n66,n70,n71,n72,n73}
87	n44	n74 (F in)	ok	n31	{n46,n47,n48,n49,n50,n51,n52,n53,n54,n55,n56,n57,n58,n59,n60,n64,n65,n66,n70,n71,n72,n73,n74}
88	n44	n60 (E out)	dead path	n31	{n46,n47,n48,n49,n50,n51,n52,n53,n54,n55,n56,n57,n58,n59,n60,n64,n65,n66,n70,n71,n72,n73,n74}
89	n46	n75 (F in)	overload	n31	{n47,n48,n49,n50,n51,n52,n53,n54,n55,n56,n57,n58,n59,n60,n64,n65,n66,n70,n71,n72,n73,n74}

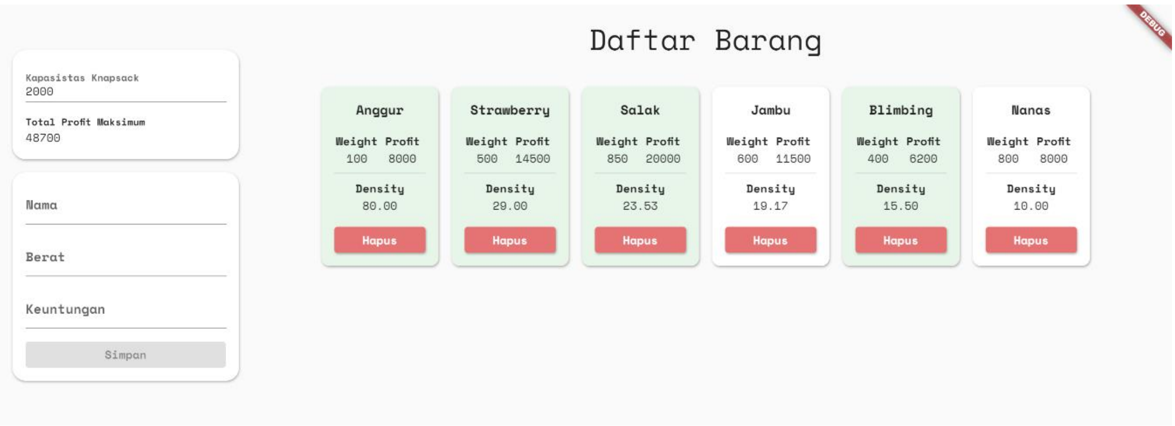
90	n46	n60 (E out)	dead path	n31	{n47,n48,n49,n50,n51,n52,n53,n54,n55,n56,n57,n58,n59,n60,n64,n65,n66,n70,n71,n72,n73,n74}
91	n47	n76 (F in)	overload	n31	{n48,n49,n50,n51,n52,n53,n54,n55,n56,n57,n58,n59,n60,n64,n65,n66,n70,n71,n72,n73,n74}
92	n47	n60 (E out)	dead path	n31	{n48,n49,n50,n51,n52,n53,n54,n55,n56,n57,n58,n59,n60,n64,n65,n66,n70,n71,n72,n73,n74}
93	n48	n77 (F in)	overload	n31	{n49,n50,n51,n52,n53,n54,n55,n56,n57,n58,n59,n60,n64,n65,n66,n70,n71,n72,n73,n74}
94	n48	n60 (E out)	dead path	n31	{n49,n50,n51,n52,n53,n54,n55,n56,n57,n58,n59,n60,n64,n65,n66,n70,n71,n72,n73,n74}
95	n49	n78 (F in)	overload	n31	{n50,n51,n52,n53,n54,n55,n56,n57,n58,n59,n60,n64,n65,n66,n70,n71,n72,n73,n74}
96	n49	n60 (E out)	dead path	n31	{n50,n51,n52,n53,n54,n55,n56,n57,n58,n59,n60,n64,n65,n66,n70,n71,n72,n73,n74}
97	n50	n79 (F in)	ok	n31	{n51,n52,n53,n54,n55,n56,n57,n58,n59,n60,n64,n65,n66,n70,n71,n72,n73,n74,n79}
98	n50	n60 (E out)	dead path	n31	{n51,n52,n53,n54,n55,n56,n57,n58,n59,n60,n64,n65,n66,n70,n71,n72,n73,n74,n79}
99	n51	n80 (F in)	ok	n31	{n52,n53,n54,n55,n56,n57,n58,n59,n60,n64,n65,n66,n70,n71,n72,n73,n74,n79,n80}
100	n51	n60 (E out)	dead path	n31	{n52,n53,n54,n55,n56,n57,n58,n59,n60,n64,n65,n66,n70,n71,n72,n73,n74,n79,n80}

101	n52	n81 (F in)	ok	n31	{n53,n54,n55,n56,n57,n58,n59,n60,n64,n65,n66,n70,n71,n72,n73,n74,n79,n80,n81}
102	n52	n60 (E out)	dead path	n31	{n53,n54,n55,n56,n57,n58,n59,n60,n64,n65,n66,n70,n71,n72,n73,n74,n79,n80,n81}
103	n53	n82 (F in)	overload	n31	{n54,n55,n56,n57,n58,n59,n60,n64,n65,n66,n70,n71,n72,n73,n74,n79,n80,n81}
104	n53	n60 (E out)	dead path	n31	{n54,n55,n56,n57,n58,n59,n60,n64,n65,n66,n70,n71,n72,n73,n74,n79,n80,n81}
105	n54	n83 (F in)	overload	n31	{n55,n56,n57,n58,n59,n60,n64,n65,n66,n70,n71,n72,n73,n74,n79,n80,n81}
106	n54	n60 (E out)	dead path	n31	{n55,n56,n57,n58,n59,n60,n64,n65,n66,n70,n71,n72,n73,n74,n79,n80,n81}
107	n55	n84 (F in)	overload	n31	{n56,n57,n58,n59,n60,n64,n65,n66,n70,n71,n72,n73,n74,n79,n80,n81}
108	n55	n60 (E out)	dead path	n31	{n56,n57,n58,n59,n60,n64,n65,n66,n70,n71,n72,n73,n74,n79,n80,n81}
109	n56	n85 (F in)	ok	n31	{n57,n58,n59,n60,n64,n65,n66,n70,n71,n72,n73,n74,n79,n80,n81,n85}
110	n56	n60 (E out)	dead path	n31	{n57,n58,n59,n60,n64,n65,n66,n70,n71,n72,n73,n74,n79,n80,n81,n85}
111	n57	n86 (F in)	ok	n31	{n58,n59,n60,n64,n65,n66,n70,n71,n72,n73,n74,n79,n80,n81,n85,n86}
112	n57	n60 (E out)	dead path	n31	{n58,n59,n60,n64,n65,n66,n70,n71,n72,n73,n74,n79,n80,n81,n85,n86}
113	n58	n87 (F in)	ok	n31	{n59,n60,n64,n65,n66,n70,n71,n72,n73,n74,n79,n80,n81,n85,n86,n87}
114	n58	n60 (E out)	dead path	n31	{n59,n60,n64,n65,n66,n70,n71,n72,n73,n74,n79,n80,n81,n85,n86,n87}

115	n59	n88 (F in)	ok	n31	{n60,n64,n65,n66,n70,n71,n72,n73,n74,n79,n80,n81,n85,n86,n87,n88}
116	n59	n60 (E out)	dead path	n31	{n60,n64,n65,n66,n70,n71,n72,n73,n74,n79,n80,n81,n85,n86,n87,n88}
117	n60	n89 (F in)	ok	n31	{n64,n65,n66,n70,n71,n72,n73,n74,n79,n80,n81,n85,n86,n87,n88,n89}
118	n60	n60 (E out)	dead path	n31	{n64,n65,n66,n70,n71,n72,n73,n74,n79,n80,n81,n85,n86,n87,n88,n89}
119	n64	-	-	n31	{n65,n66,n70,n71,n72,n73,n74,n79,n80,n81,n85,n86,n87,n88,n89}
120	n65	-	-	n31	{n66,n70,n71,n72,n73,n74,n79,n80,n81,n85,n86,n87,n88,n89}
121	n66	-	-	n31	{n70,n71,n72,n73,n74,n79,n80,n81,n85,n86,n87,n88,n89}
122	n70	-	-	n31	{n71,n72,n73,n74,n79,n80,n81,n85,n86,n87,n88,n89}
123	n71	-	-	n31	{n72,n73,n74,n79,n80,n81,n85,n86,n87,n88,n89}
124	n72	-	-	n31	{n73,n74,n79,n80,n81,n85,n86,n87,n88,n89}
125	n73	-	-	n31	{n74,n79,n80,n81,n85,n86,n87,n88,n89}
126	n74	-	-	n31	{n79,n80,n81,n85,n86,n87,n88,n89}
127	n79	-	-	n31	{n80,n81,n85,n86,n87,n88,n89}
128	n80	-	-	n31	{n81,n85,n86,n87,n88,n89}
129	n81	-	-	n31	{n85,n86,n87,n88,n89}
130	n85	-	-	n31	{n86,n87,n88,n89}
131	n86	-	-	n31	{n87,n88,n89}
132	n87	-	-	n31	{n88,n89}
133	n88	-	-	n31	{n89}
134	n89	-	-	n31	{}

Solusi berakhir pada simpul yang memiliki total profit tertinggi yaitu n31 dengan nilai total profit 48700. Sehingga didapatkan hasil sebagai berikut.

Label	**Nama**	**Weight**	**Profit**	**Density**	**Ambil**
A	Anggur	100.0	8000.0	80.0	Ya
B	Strawberry	500.0	14500.0	29.0	Ya
C	Salak	850.0	20000.0	~23.52	Ya
D	Jambu	600.0	11500.0	~19.16	Tidak
E	Blimbing	400.0	6200.0	15.5	Ya
F	Nanas	800.0	8000.0	10.0	Tidak



Daftar Barang

Kapasistas Knapsack
2000

Total Profit Maksimum
48700

Nama

Berat

Keuntungan

Anggur	Strawberry	Salak	Jambu	Blimbing	Nanas
Weight Profit 100 8000	Weight Profit 500 14500	Weight Profit 850 20000	Weight Profit 600 11500	Weight Profit 400 6200	Weight Profit 800 8000
Density 80.00	Density 29.00	Density 23.53	Density 19.17	Density 15.50	Density 10.00

F. Perhitungan dataset 6

Kapasitas keranjang = 1000

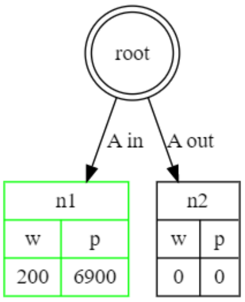
Nama	**Weight**	**Profit**	**Density**
Jambu	600.0	11500.0	~19.16
Kurma	800.0	12000.0	15.0
Nanas	800.0	8000.0	10.0
Apel	200.0	6900.0	34.5

Lakukan pengurutan berdasarkan *density* secara *descending* untuk mempermudah pencarian berdasarkan *greedy by density*.

Label	**Nama**	**Weight**	**Profit**	**Density**
A	Apel	200.0	6900.0	34.5
B	Jambu	600.0	11500.0	~19.16
C	Kurma	800.0	12000.0	15.0

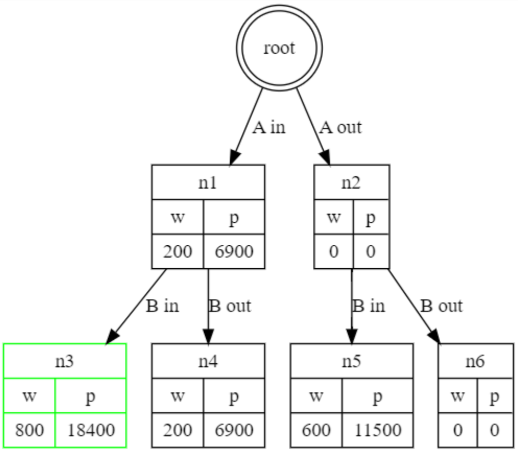
D	Nanas	800.0	8000.0	10.0

Langkah pada *depth* 0



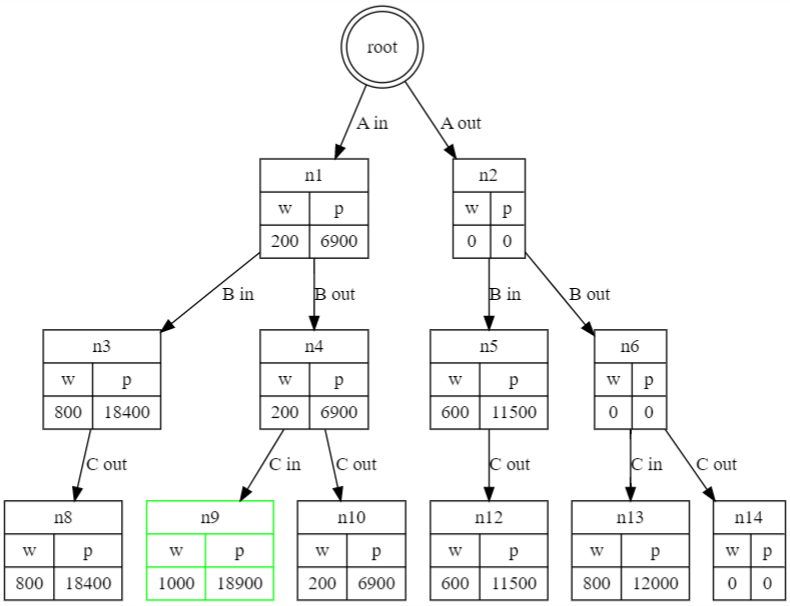
itr	current	branch	status	solution	queue
1	root	n1 (A in)	ok	n1	{n1}
2	root	n2 (A out)	ok	n1	{n1,n2}

Langkah pada *depth* 1



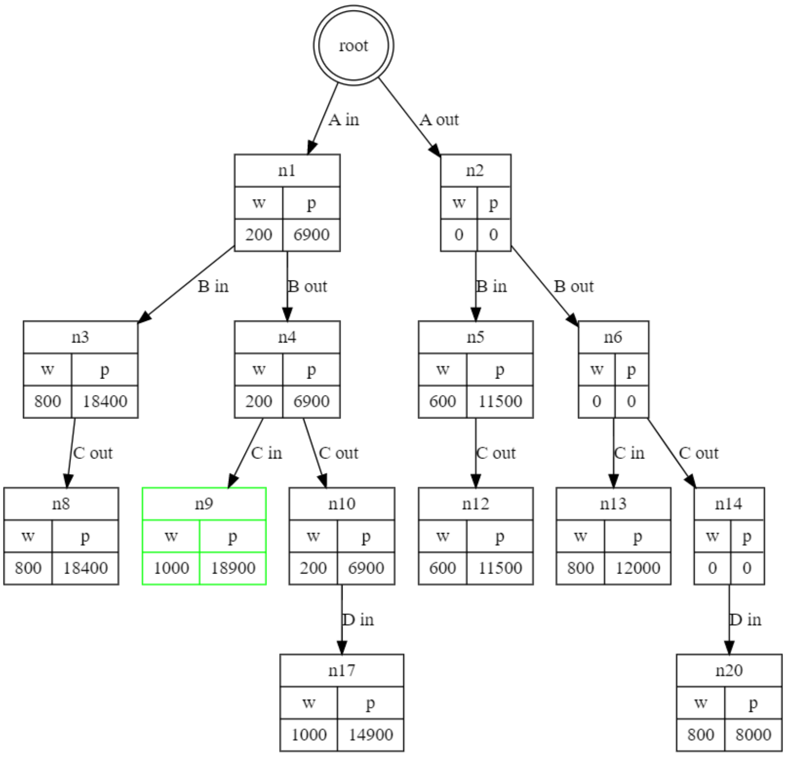
itr	current	branch	status	solution	queue
3	n1	n3 (B in)	ok	n3	{n2,n3}
4	n1	n4 (B out)	ok	n3	{n2,n3,n4}
5	n2	n5 (B in)	ok	n3	{n3,n4,n5}
6	n2	n6 (B out)	ok	n3	{n3,n4,n5,n6}

Langkah pada *depth* 2



itr	current	branch	status	solution	queue
7	n3	n7 (C in)	overload	n3	{n4,n5,n6}
8	n3	n8 (C out)	ok	n8	{n4,n5,n6,n8}
9	n4	n9 (C in)	ok	n9	{n5,n6,n8,n9}
10	n4	n10 (C out)	ok	n9	{n5,n6,n8,n9,n10}
11	n5	n11 (C in)	overload	n9	{n6,n8,n9,n10}
12	n5	n12 (C out)	ok	n9	{n6,n8,n9,n10,n12}
13	n6	n13 (C in)	ok	n9	{n8,n9,n10,n12,n13}
14	n6	n14 (C out)	ok	n9	{n8,n9,n10,n12,n13,n14}

Langkah pada *depth* 3

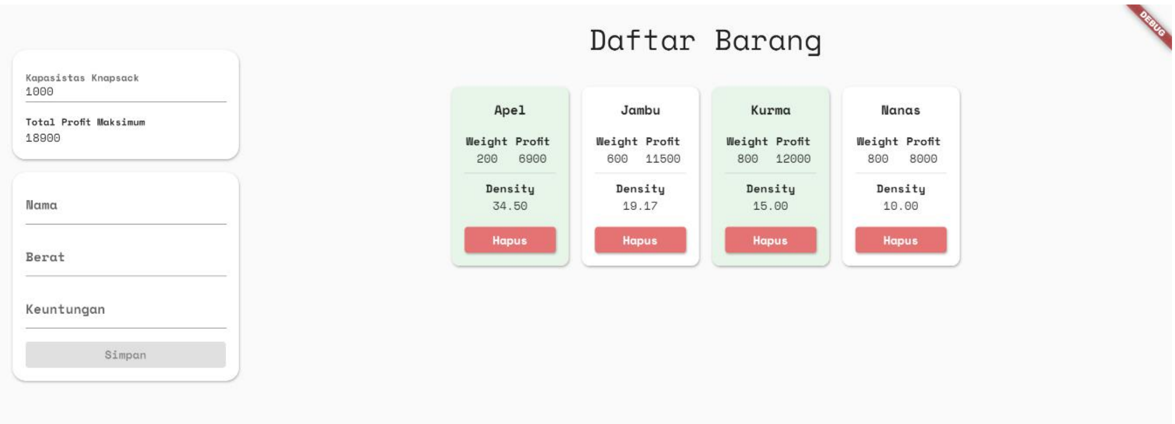


itr	current	branch	status	solution	queue
15	n8	n15 (D in)	overload	n9	{n9,n10,n12,n13,n14}
16	n8	n14 (C out)	dead path	n9	{n9,n10,n12,n13,n14}
17	n9	n16 (D in)	overload	n9	{n10,n12,n13,n14}
18	n9	n14 (C out)	dead path	n9	{n10,n12,n13,n14}
19	n10	n17 (D in)	ok	n9	{n12,n13,n14,n17}
20	n10	n14 (C out)	dead path	n9	{n12,n13,n14,n17}
21	n12	n18 (D in)	overload	n9	{n13,n14,n17}
22	n12	n14 (C out)	dead path	n9	{n13,n14,n17}
23	n13	n19 (D in)	overload	n9	{n14,n17}
24	n13	n14 (C out)	dead path	n9	{n14,n17}
25	n14	n20 (D in)	ok	n9	{n17,n20}
26	n14	n14 (C out)	dead path	n9	{n17,n20}
27	n17	-	-	n9	{n20}
28	n20	-	-	n9	{ }

Solusi berakhir pada simpul yang memiliki total profit tertinggi yaitu n9 dengan nilai total profit 18900. Sehingga didapatkan hasil sebagai berikut.

Label	**Nama**	**Weight**	**Profit**	**Density**	**Ambil**

A	Apel	200.0	6900.0	34.5	Ya
B	Jambu	600.0	11500.0	~19.16	Tidak
C	Kurma	800.0	12000.0	15.0	Ya
D	Nanas	800.0	8000.0	10.0	Tidak


Daftar Barang

Kapasistas Knapsack
1000

Total Profit Maksimum
18900

Nama
Berat
Keuntungan

Apel
Weight Profit
200 6900
Density
34.50

Jambu
Weight Profit
600 11500
Density
19.17

Kurma
Weight Profit
800 12000
Density
15.00

Nanas
Weight Profit
800 8000
Density
10.00

G. Perhitungan dataset 7

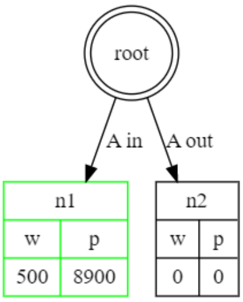
Kapasitas keranjang = 1500

Nama	Weight	Profit	Density
Blimbing	400.0	6200.0	15.5
Rambutan	500.0	8900.0	17.8
Kurma	800.0	12000.0	15.0
Jeruk	250.0	3800.0	15.2

Lakukan pengurutan berdasarkan *density* secara *descending* untuk mempermudah pencarian berdasarkan *greedy by density*.

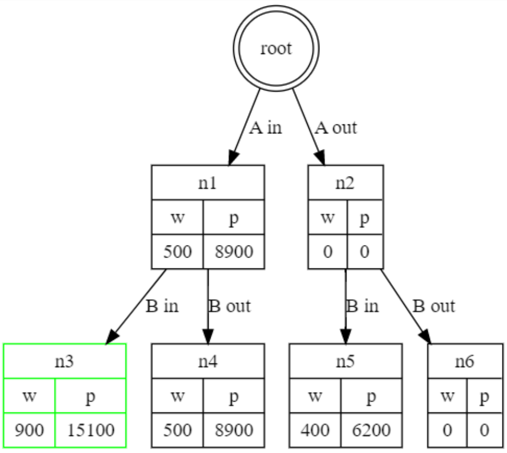
Label	Nama	Weight	Profit	Density
A	Rambutan	500.0	8900.0	17.8
B	Blimbing	400.0	6200.0	15.5
C	Jeruk	250.0	3800.0	15.2
D	Kurma	800.0	12000.0	15.0

Langkah pada *depth* 0



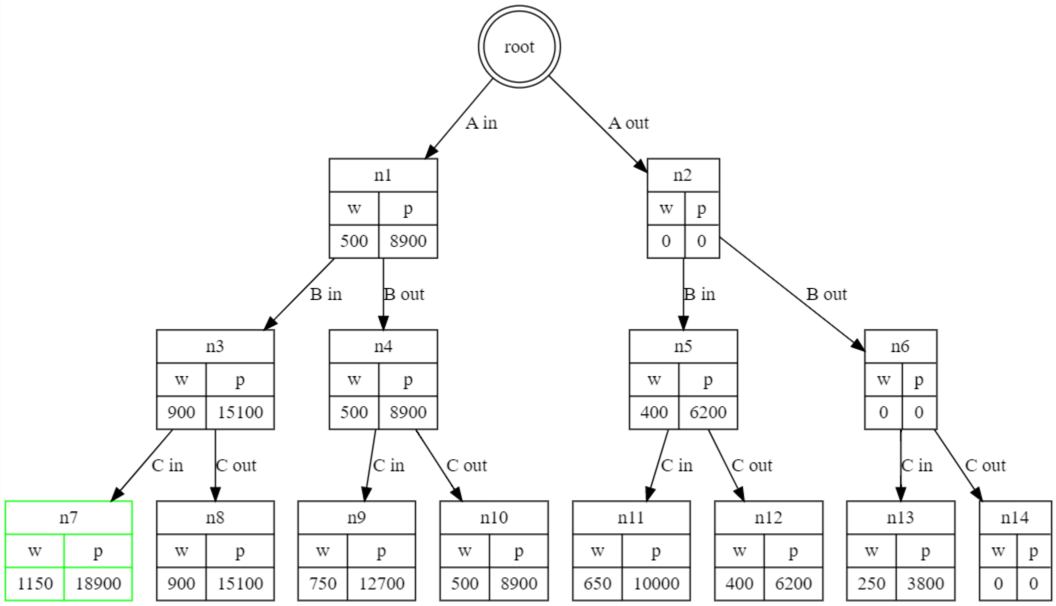
itr	current	branch	status	solution	queue
1	root	n1 (A in)	ok	n1	{n1}
2	root	n2 (A out)	ok	n1	{n1,n2}

Langkah pada *depth* 1



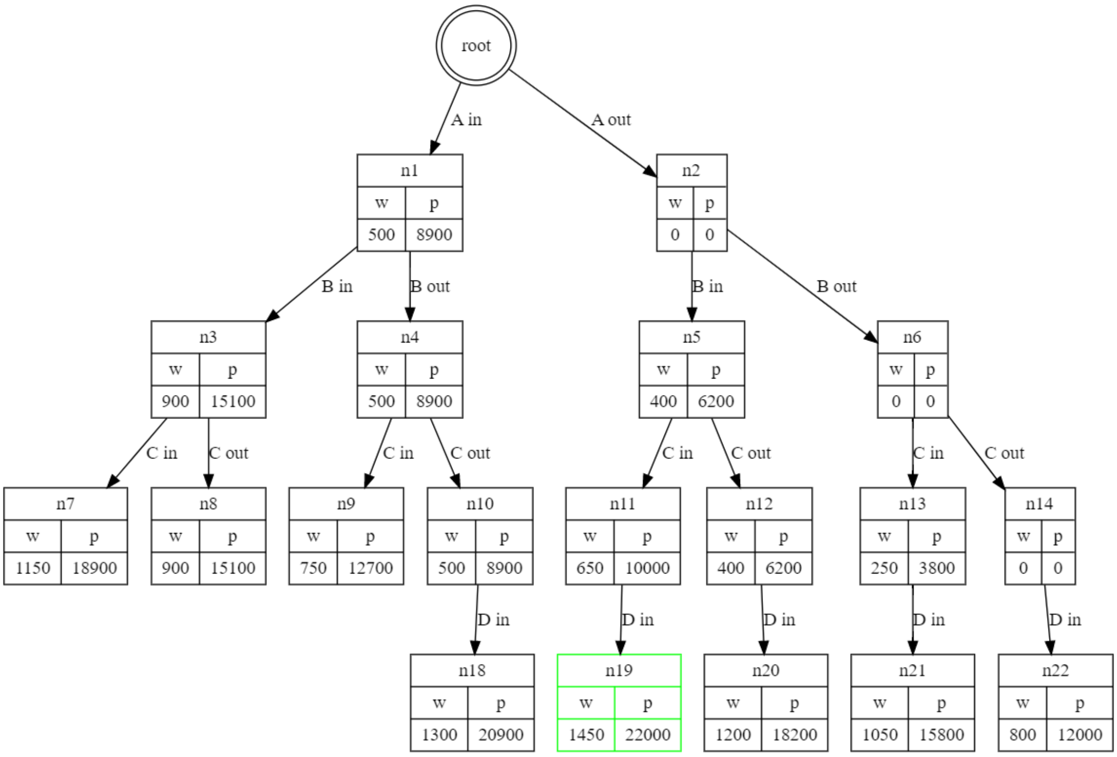
itr	current	branch	status	solution	queue
3	n1	n3 (B in)	ok	n3	{n2,n3}
4	n1	n4 (B out)	ok	n3	{n2,n3,n4}
5	n2	n5 (B in)	ok	n3	{n3,n4,n5}
6	n2	n6 (B out)	ok	n3	{n3,n4,n5,n6}

Langkah pada *depth* 2



itr	current	branch	status	solution	queue
7	n3	n7 (C in)	ok	n7	{n4,n5,n6,n7}
8	n3	n8 (C out)	ok	n7	{n4,n5,n6,n7,n8}
9	n4	n9 (C in)	ok	n7	{n5,n6,n7,n8,n9}
10	n4	n10 (C out)	ok	n7	{n5,n6,n7,n8,n9,n10}
11	n5	n11 (C in)	ok	n7	{n6,n7,n8,n9,n10,n11}
12	n5	n12 (C out)	ok	n7	{n6,n7,n8,n9,n10,n11,n12}
13	n6	n13 (C in)	ok	n7	{n7,n8,n9,n10,n11,n12,n13}
14	n6	n14 (C out)	ok	n7	{n7,n8,n9,n10,n11,n12,n13,n14}

Langkah pada *depth* 3

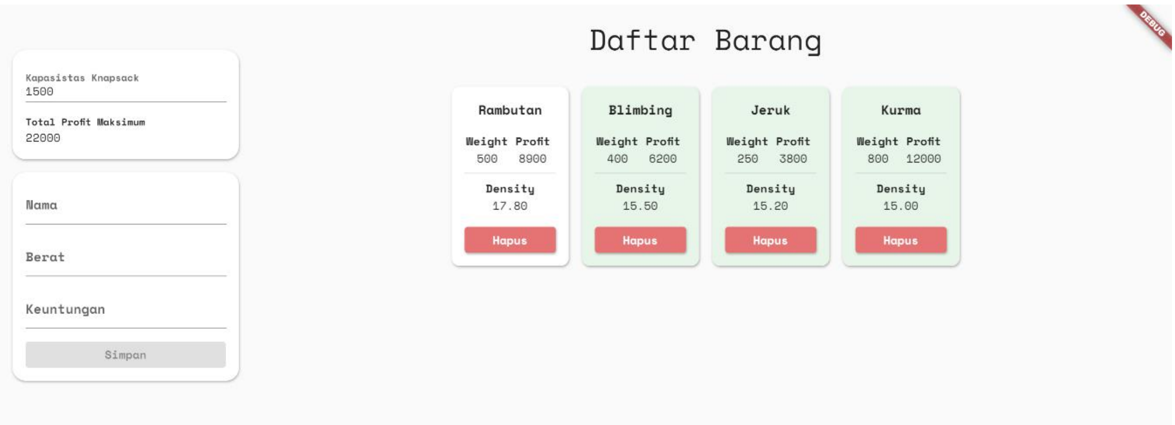


itr	current	branch	status	solution	queue
15	n7	n15 (D in)	overload	n7	{n8,n9,n10,n11,n12,n13,n14}
16	n7	n14 (C out)	dead path	n7	{n8,n9,n10,n11,n12,n13,n14}
17	n8	n16 (D in)	overload	n7	{n9,n10,n11,n12,n13,n14}
18	n8	n14 (C out)	dead path	n7	{n9,n10,n11,n12,n13,n14}
19	n9	n17 (D in)	overload	n7	{n10,n11,n12,n13,n14}
20	n9	n14 (C out)	dead path	n7	{n10,n11,n12,n13,n14}
21	n10	n18 (D in)	ok	n18	{n11,n12,n13,n14,n18}
22	n10	n14 (C out)	dead path	n18	{n11,n12,n13,n14,n18}
23	n11	n19 (D in)	ok	n19	{n12,n13,n14,n18,n19}
24	n11	n14 (C out)	dead path	n19	{n12,n13,n14,n18,n19}
25	n12	n20 (D in)	ok	n19	{n13,n14,n18,n19,n20}
26	n12	n14 (C out)	dead path	n19	{n13,n14,n18,n19,n20}
27	n13	n21 (D in)	ok	n19	{n14,n18,n19,n20,n21}
28	n13	n14 (C out)	dead path	n19	{n14,n18,n19,n20,n21}
29	n14	n22 (D in)	ok	n19	{n18,n19,n20,n21,n22}
30	n14	n14 (C out)	dead path	n19	{n18,n19,n20,n21,n22}
31	n18	-	-	n19	{n19,n20,n21,n22}
32	n19	-	-	n19	{n20,n21,n22}

33	n20	-	-	n19	{n21,n22}
34	n21	-	-	n19	{n22}
35	n22	-	-	n19	{ }

Solusi berakhir pada simpul yang memiliki total profit tertinggi yaitu n19 dengan nilai total profit 22000. Sehingga didapatkan hasil sebagai berikut.

Label	**Nama**	**Weight**	**Profit**	**Density**	**Ambil**
A	Rambutan	500.0	8900.0	17.8	Tidak
B	Blimbing	400.0	6200.0	15.5	Ya
C	Jeruk	250.0	3800.0	15.2	Ya
D	Kurma	800.0	12000.0	15.0	Ya



Daftar Barang

Kapasistas Knapsack
1500

Total Profit Maksimum
22000

Nama

Berat

Keuntungan

Rambutan	Blimbing	Jeruk	Kurma
Weight Profit 500 8900	Weight Profit 400 6200	Weight Profit 250 3800	Weight Profit 800 12000
Density 17.80	Density 15.50	Density 15.20	Density 15.00

Kode program pada setiap file dipaparkan di bawah ini.

app_provider.dart

```dart
import 'dart:async';
import 'dart:collection';

import 'package:flutter/material.dart';

class AppProvider extends ChangeNotifier {
  AppProvider() {
    int index = 1;
    switch (index) {
      case 1:
        _maxWeight = 1500;
        items = [
          Item("Jambu", 11500, 600),
          Item("Kiwi", 18000, 500),
          Item("Lemon", 6000, 450),
          Item("Buah Naga", 6200, 400),
          Item("Apel", 6900, 200),
        ];
        break;
      case 2:
        _maxWeight = 1750;
        items = [
          Item("Delima", 5500, 800),
          Item("Salak", 20000, 850),
          Item("Buah Naga", 6200, 400),
          Item("Srikaya", 3500, 500),
          Item("Markisa", 5000, 600),
        ];
        break;
      case 3:
        _maxWeight = 1600;
        items = [
          Item("Salak", 20000, 850),
          Item("Delima", 5500, 800),
          Item("Buah Naga", 6200, 400),
          Item("Nanas", 8000, 800),
          Item("Mangga", 10000, 750),
          Item("Blimbing", 6200, 400),
        ];
        break;
      case 4:
        _maxWeight = 1400;
        items = [
          Item("Buah Naga", 6200, 400),
          Item("Anggur", 8000, 100),
          Item("Pisang", 12000, 700),
          Item("Kurma", 12000, 800),
          Item("Jeruk", 3800, 250),
          Item("Kelengkeng", 13300, 750),
        ];
        break;
      case 5:
        _maxWeight = 2000;
        items = [
          Item("Blimbing", 6200, 400),
          Item("Strawberry", 14500, 500),
        Item("Jambu", 11500, 600),
        Item("Anggur", 8000, 100),
        Item("Salak", 20000, 850),
        Item("Nanas", 8000, 800),
      ];
      break;
    case 6:
      _maxWeight = 1000;
      items = [
        Item("Jambu", 11500, 600),
        Item("Kurma", 12000, 800),
        Item("Nanas", 8000, 800),
        Item("Apel", 6900, 200),
      ];
      break;
    case 7:
      _maxWeight = 1500;
      items = [
        Item("Blimbing", 6200, 400),
        Item("Rambutan", 8900, 500),
        Item("Kurma", 12000, 800),
        Item("Jeruk", 3800, 250),
      ];
      break;
    default:
      _maxWeight = 0;
      items = [];
      break;
  }

  compute();
}

bool get isInputValid => _name.isNotEmpty && _weight > 0 && _profit > 0;

String _name = "";
String get name => _name;
set name(String name) {
  _name = name;
  notifyListeners();
}

double _weight = 0;
double get weight => _weight;
set weight(double weight) {
  _weight = weight;
  notifyListeners();
}

double _profit = 0;
double get profit => _profit;
set profit(double profit) {
  _profit = profit;
  notifyListeners();
}

late double _maxWeight;
double get maxWeight => _maxWeight;
set maxWeight(double maxWeight) {
  _maxWeight = maxWeight;
  compute();
```

```
}

late List<Item> items;

void add() {
  if (isInputValid) {
    items.add(Item(name, profit, weight));

    compute();

    _profit = 0;
    _weight = 0;

    notifyListeners();
  }
}

void delete(int i) {
  items.removeAt(i);
  compute();
  notifyListeners();
}

double totalProfit = 0;

Timer? _debounce;
void compute() {
  if (_debounce != null && _debounce!.isActive) _debounce!.cancel();

  _debounce = Timer(const Duration(milliseconds: 500), () {
    Node result = knapsack(maxWeight, items);

    totalProfit = result.totalProfit;
    for (var item in items) {
      item.isSelected = false;
    }

    Node? temp = result;
    while (temp != null) {
      if (temp.level >= 0) {
        items[temp.level].isSelected = temp.status;
        print(temp);
      }

      temp = temp.parent;
    }

    notifyListeners();
  });
}

Node knapsack(double maxWeight, List<Item> items) {
  // Pengurutan density secara descending
  items.sort((a, b) => b.density.compareTo(a.density));

  // Membuat queue untuk penjelajahan graf
  Node u = Node(level: -1);
  Node v1 = Node();
  Node v2 = Node();
  Queue<Node> queue = Queue.from([u]);
```

```
    // Proses komputasi brute force BFS
    Node maxNode = u;

    while (queue.isNotEmpty) {
      // Dequeue node
      u = queue.removeFirst();

      // Jika tidak punya cabang maka skip
      if (u.level < items.length - 1) {
        // Ambil node
        v1 = Node(
          level: u.level + 1,
          totalWeight: u.totalWeight + items[u.level + 1].weight,
          totalProfit: u.totalProfit + items[u.level + 1].profit,
          parent: u,
          status: true,
        );

        // Jika profit baru kurang dari W dan profit lebih besar dari sebelumnya
        // Update profit
        if (v1.totalWeight <= maxWeight) {
          queue.addLast(v1);

          if (v1.totalProfit >= maxNode.totalProfit) {
            maxNode = v1;
          }
        }

        if (u.level < items.length - 2) {
          // Tidak mengambil node
          v2 = Node(
            level: u.level + 1,
            totalWeight: u.totalWeight,
            totalProfit: u.totalProfit,
            parent: u,
          );

          queue.addLast(v2);

          if (v2.totalProfit >= maxNode.totalProfit) {
            maxNode = v2;
          }
        }
      }
    }

    return maxNode;
  }
}

class Item {
  bool isSelected = false;
  final String name;
  final double profit;
  final double weight;

  Item(this.name, this.profit, this.weight);

  double get density => profit / weight;

  @override
  String toString() => "Item(name: $name, profit: $profit, weight: $weight)";
}

class Node {
  final Node? parent;
  final bool status;
  final double totalProfit;
  final double totalWeight;
  final int level;

  Node({
    this.status = false,
    this.level = 0,
    this.parent,
    this.totalProfit = 0,
    this.totalWeight = 0,
  });

  @override
  String toString() =>
      "Node(level: $level, profit: ${totalProfit}, weight: ${totalWeight},
status: $status)";
}
```

dashboard.dart

```
import 'package:flutter/material.dart';
import 'package:google_fonts/google_fonts.dart';
import 'package:knapsack_gui/app_provider.dart';
import 'package:provider/provider.dart';

class DashboardPage extends StatelessWidget {
  const DashboardPage({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    TextEditingController nameController = TextEditingController();
    TextEditingController weightController = TextEditingController();
    TextEditingController profitController = TextEditingController();

    return ChangeNotifierProvider<AppProvider>(
      create: (context) => AppProvider(),
      builder: (context, _) => Scaffold(
        body: Container(
          padding: const EdgeInsets.all(16),
          width: MediaQuery.of(context).size.width,
          height: MediaQuery.of(context).size.height,
          child: Row(
            children: [
              Expanded(
                flex: 2,
                child: Column(
                  mainAxisAlignment: MainAxisAlignment.center,
                  children: [
                    Material(
                      color: Colors.white,
                      elevation: 2,
                      borderRadius: BorderRadius.circular(16),
                      child: Padding(
                        padding: const EdgeInsets.all(16),
                        child: Column(
                          crossAxisAlignment: CrossAxisAlignment.start,
                          children: [
                            TextFormField(
                              keyboardType: TextInputType.number,
                              style: GoogleFonts.spaceMono(fontSize: 14),
                              decoration: InputDecoration(
                                labelStyle: GoogleFonts.spaceMono(
                                  fontSize: 16,
                                  fontWeight: FontWeight.w600,
                                ),
                                labelText: "Kapasistas Knapsack",
                              ),
                              initialValue: Provider.of<AppProvider>(context,
                                      listen: false)
                                  .maxWeight
                                  .toString(),
                              onChanged: (value) {
                                if (value.isEmpty) {
                                  Provider.of<AppProvider>(context,
                                          listen: false)
                                      .maxWeight = 0;
                                  return;
                                }

                                try {
                                  Provider.of<AppProvider>(context,
                                          listen: false)
                                      .maxWeight = double.parse(value);
                                } catch (e) {
                                  // do nothing
                                }
                              },
                            ),
                            const SizedBox(height: 16),
                            Text(
                              "Total Profit Maksimum",
                              style: GoogleFonts.spaceMono(
                                fontSize: 12,
                                fontWeight: FontWeight.w600,
                              ),
                            ),
                            Consumer<AppProvider>(
                              builder: (context, state, _) => Text(
                                state.totalProfit.toString(),
                                style: GoogleFonts.spaceMono(fontSize: 14),
                              ),
                            ),
                          ],
                        ),
                      ),
                    ),
                    const SizedBox(height: 16),
                    Material(
                      color: Colors.white,
                      elevation: 2,
                      borderRadius: BorderRadius.circular(16),
                      child: Padding(
                        padding: const EdgeInsets.all(16),
                        child: Column(
                    children: [
                      TextFormField(
                        controller: nameController,
                        keyboardType: TextInputType.text,
                        textInputAction: TextInputAction.next,
                        style: GoogleFonts.spaceMono(fontSize: 14),
                        decoration: InputDecoration(
                          labelStyle: GoogleFonts.spaceMono(
                            fontSize: 16,
                            fontWeight: FontWeight.w600,
                          ),
                          labelText: "Nama",
                        ),
                        onChanged: (value) {
                          if (value.isEmpty) {
                            Provider.of<AppProvider>(context,
                                    listen: false)
                                .name = "";
                            return;
                          }

                          try {
                            Provider.of<AppProvider>(context,
                                    listen: false)
                                .name = value;
                          } catch (e) {
                            // do nothing
                          }
                        },
                      ),
                      const SizedBox(height: 16),
                      TextFormField(
                        controller: weightController,
                        keyboardType: TextInputType.number,
                        textInputAction: TextInputAction.next,
                        style: GoogleFonts.spaceMono(fontSize: 14),
                        decoration: InputDecoration(
                          labelStyle: GoogleFonts.spaceMono(
                            fontSize: 16,
                            fontWeight: FontWeight.w600,
                          ),
                          labelText: "Berat",
                        ),
                        onChanged: (value) {
                          if (value.isEmpty) {
                            Provider.of<AppProvider>(context,
                                    listen: false)
                                .weight = 0;
                            return;
                          }

                          try {
                            Provider.of<AppProvider>(context,
                                    listen: false)
                                .weight = double.parse(value);
                          } catch (e) {
                            // do nothing
                          }
                        },
                      ),
                      const SizedBox(height: 16),
                          TextFormField(
                            controller: profitController,
                            keyboardType: TextInputType.number,
                            style: GoogleFonts.spaceMono(fontSize: 14),
                            decoration: InputDecoration(
                              labelStyle: GoogleFonts.spaceMono(
                                fontSize: 16,
                                fontWeight: FontWeight.w600,
                              ),
                              labelText: "Keuntungan",
                            ),
                            onChanged: (value) {
                              if (value.isEmpty) {
                                Provider.of<AppProvider>(context,
                                        listen: false)
                                    .profit = 0;
                                return;
                              }

                              try {
                                Provider.of<AppProvider>(context,
                                        listen: false)
                                    .profit = double.parse(value);
                              } catch (e) {
                                // do nothing
                              }
                            },
                          ),
                          const SizedBox(height: 16),
                          SizedBox(
                            width: double.infinity,
                            child: Consumer<AppProvider>(
                              child: Text(
                                "Simpan",
                                style: GoogleFonts.spaceMono(
                                  fontSize: 14,
                                  fontWeight: FontWeight.w600,
                                ),
                              ),
                              builder: (context, state, child) =>
                                  ElevatedButton(
                                onPressed: state.isInputValid
                                    ? () {
                                        nameController.text = "";
                                        weightController.text = "";
                                        profitController.text = "";
                                        Provider.of<AppProvider>(context,
                                                listen: false)
                                            .add();
                                      }
                                    : null,
                                child: child,
                                style: ElevatedButton.styleFrom(
                                  visualDensity: VisualDensity.comfortable,
                                ),
                              ),
                            ),
                          ),
                        ],
                      ),
                    ),
                                        ),
                                      ],
                                    ),
                                  ),
                                  const SizedBox(width: 16),
                                  Expanded(
                                    flex: 8,
                                    child: Column(
                                      children: [
                                        Text(
                                          "Daftar Barang",
                                          style: GoogleFonts.spaceMono(
                                            fontSize: 36,
                                          ),
                                        ),
                                        const SizedBox(height: 16),
                                        Expanded(
                                          child: SingleChildScrollView(
                                            padding: const EdgeInsets.all(16),
                                            child: Consumer<AppProvider>(
                                              builder: (context, state, _) => Wrap(
                                                runSpacing: 16,
                                                spacing: 16,
                                                alignment: WrapAlignment.center,
                                                children: List.generate(
                                                  state.items.length,
                                                  (i) => ItemCard(
                                                    item: state.items[i],
                                                    onTap: () {
                                                      state.delete(i);
                                                    },
                                                  ),
                                                ),
                                              ),
                                            ),
                                          ),
                                        ),
                                      ],
                                    ),
                                  ),
                                ],
                              ),
                            ),
                          ),
                        );
                      }
                    }

                    class ItemCard extends StatelessWidget {
                      final Item item;
                      final void Function()? onTap;

                      const ItemCard({
                        Key? key,
                        required this.item,
                        this.onTap,
                      }) : super(key: key);

                      @override
                      Widget build(BuildContext context) {
                        return Material(
                    color: item.isSelected ? Colors.green.shade50 : Colors.white,
                    elevation: 2,
                    borderRadius: BorderRadius.circular(10),
                    child: Container(
                      padding: const EdgeInsets.all(16),
                      width: MediaQuery.of(context).size.width / 10,
                      child: Column(
                        children: [
                          Text(
                            item.name,
                            textAlign: TextAlign.center,
                            style: GoogleFonts.spaceMono(
                              fontSize: 16,
                              fontWeight: FontWeight.w900,
                            ),
                          ),
                          const SizedBox(height: 16),
                          Row(
                            children: [
                              Expanded(
                                child: Column(
                                  children: [
                                    Text(
                                      "Weight",
                                      style: GoogleFonts.spaceMono(
                                        fontSize: 14,
                                        fontWeight: FontWeight.w600,
                                      ),
                                    ),
                                    Text(
                                      "${item.weight}",
                                      style: GoogleFonts.spaceMono(fontSize: 14),
                                    ),
                                  ],
                                ),
                              ),
                              Expanded(
                                child: Column(
                                  children: [
                                    Text(
                                      "Profit",
                                      style: GoogleFonts.spaceMono(
                                        fontSize: 14,
                                        fontWeight: FontWeight.w600,
                                      ),
                                    ),
                                    Text(
                                      "${item.profit}",
                                      style: GoogleFonts.spaceMono(fontSize: 14),
                                    ),
                                  ],
                                ),
                              ),
                            ],
                          ),
                          const Divider(thickness: 1),
                          Column(
                            children: [
                              Text(
                                "Density",
                                style: GoogleFonts.spaceMono(
                                        fontSize: 14,
                                        fontWeight: FontWeight.w600,
                                      ),
                                    ),
                                    Text(
                                      item.density.toStringAsFixed(2),
                                      style: GoogleFonts.spaceMono(fontSize: 14),
                                    ),
                                  ],
                                ),
                                const SizedBox(height: 16),
                                SizedBox(
                                  width: double.infinity,
                                  child: ElevatedButton(
                                    onPressed: onTap,
                                    child: Text(
                                      "Hapus",
                                      style: GoogleFonts.spaceMono(
                                        fontSize: 14,
                                        fontWeight: FontWeight.w600,
                                      ),
                                    ),
                                    style: ElevatedButton.styleFrom(
                                      primary: Colors.red.shade300,
                                      visualDensity: VisualDensity.comfortable,
                                    ),
                                  ),
                                ),
                              ],
                            ),
                          ),
                        );
                      }
                    }
```

main.dart

```
import 'package:flutter/material.dart';
import 'package:knapsack_gui/dashboard.dart';

void main() {
  runApp(const MyApp());
}

class MyApp extends StatelessWidget {
  const MyApp({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Knapsack GUI',
      theme: ThemeData(
        primarySwatch: Colors.blue,
      ),
      home: DashboardPage(),
    );
  }
}
```