Penggunaan Pohon Biner untuk Memecahkan Persoalan *Two Dimensional Bin Packing*

Chiquita Ahsanunnisa - 13521129[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]13521129@mahasiswa.itb.ac.id

***Abstract* — *Two dimensional bin packing problem membahas* bagaimana caranya menyusun balok-balok ke dalam *bin* dua dimensi secara optimal?" Persoalan ini dikategorikan sebagai *nondeterministic polynomial* (NP) *problem* yang membutuhkan algoritma kompleks untuk memecahkannya. Oleh karena itu, dibutuhkan suatu algoritma pendekatan. Salah satu pendekatan yang dapat dilakukan adalah dengan memanfaatkan konsep pohon biner. Meskipun demikian, pendekatan dengan konsep pohon biner ini juga memiliki kekurangan.**

***Keywords*— Bin Packing, Pohon, Pohon Biner, Two Dimensional Bin Packing.**

I. PENDAHULUAN

Two dimensional bin packing problem adalah persoalan algoritma klasik. Inti dari permasalahan ini adalah "bagaimana caranya menyusun balok-balok ke dalam *bin* dua dimensi secara optimal?" Meskipun terdengar simpel, aplikasi dari persoalan ini cukup krusial. Pemecahan dari permasalahan ini banyak diaplikasikan pada industri material (kaca, logam, kayu) untuk mengoptimalkan material yang dipotong pada proses *cutting*. Selain itu, persoalan ini juga diaplikasikan pada bidang transportasi, *warehousing*, dan desain *web*.

Namun, terlepas dari kekrusialannya, persoalan ini justru dikategorikan sebagai *nondeterministic polynomial* (NP) yang membutuhkan algoritma kompleks untuk memecahkannya. Oleh karena itu, dibutuhkan algoritma pendekatan (*approximation algorithms*) untuk memecahkan persoalan ini.

Salah satu pendekatan yang dapat digunakan untuk memecahkan persoalan ini adalah dengan memanfaatkan konsep pohon biner. Pada makalah ini, dibahas bagaimana konsep pohon biner dapat digunakan untuk menyelesaikan 2D *bin packing problem*. Tak hanya membahas dengan tulisan saja, penulis juga menyediakan media simulasi untuk memvisualisasikan 2D *bin packing problem*.

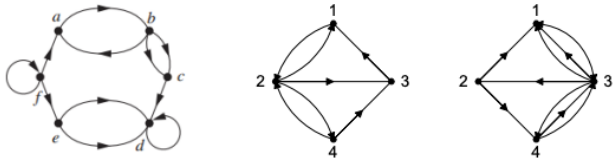
II. LANDASAN TEORI

A. Graf

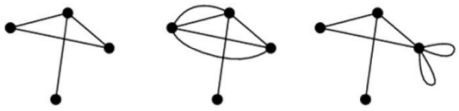
Graf adalah struktur data yang digunakan untuk merepresentasikan hubungan antara objek-objek diskrit. Secara formal, sebuah graf didefinisikan dengan *tuple* (*V, E*), dengan *V* adalah himpunan tidak kosong dari simpul-simpul (*vertices*) dan *E* adalah himpunan sisi-sisi (*edges*) yang menghubungkan sepasang simpul.

Berdasarkan orientasi arah pada sisi graf, graf dibagi menjadi dua jenis, yaitu sebagai berikut.

1. Graf berarah
   Sisi pada graf ini mempunyai orientasi arah yang bersifat *one-directional*. Sisi pada graf ini divisualisasikan dengan garis berpanah.
2. Graf tak berarah
   Sisi pada graf ini tidak mempunyai orientasi arah. Artinya, hubungan yang dinyatakan dengan sisi tersebut bersifat mutualisme. Sisi pada graf ini divisualisasikan dengan garis saja.



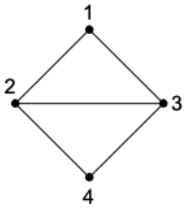
(a) Graf berarah



(b) Graf tidak berarah

Gambar 1 Jenis Graf berdasarkan Orientasi Arah pada Sisi
Sumber: *https://informatika.stei.itb.ac.id/~rinaldi.munir/*

Pada graf, dikenal istilah lintasan (*path*) dan sirkuit (*cycle*). Lintasan yang panjangnya *n* dari simpul awal v_0 ke simpul tujuan v_n di dalam graf *G* ialah barisan berselang-seling simpul-simpul dan sisi-sisi yang berbentuk $v_0, e_1, v_1, e_2, v_2, ..., v_{n-1}, e_n, v_n$ sedemikian sehingga $e_1 = (v_0, v_1)$, $e_2 = (v_1, v_2)$, ..., $e_n = (v_{n-1}, v_n)$ adalah sisi-sisi dari graf *G*. Sirkuit adalah lintasan yang berawal dan berakhir pada simpul yang sama.



Gambar 2 Graf G_1
Sumber: *https://informatika.stei.itb.ac.id/~rinaldi.munir/*

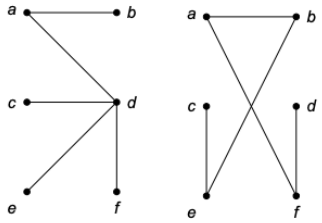
Pada graf G_1 di atas, jalur 1-3-4 adalah sebuah lintasan, sedangkan jalur 1-3-4-2-1 adalah sebuah sirkuit.

B. Pohon

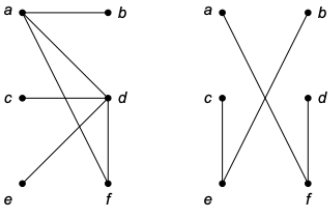
Pohon adalah graf tidak berarah yang terhubung dan tidak mengandung sirkuit. Definisi lain yang lebih formal dari pohon adalah sebagai berikut.

Jika $G = (V, E)$ adalah graf tak-berarah sederhana dan jumlah simpulnya n, semua pernyataan di bawah ini ekivalen:

1. G adalah pohon.
2. Setiap pasang simpul di dalam G terhubung dengan lintasan tunggal.
3. G terhubung dan memiliki $m = n - 1$ buah sisi.
4. G tidak mengandung sirkuit dan memiliki $m = n - 1$ buah sisi.
5. G tidak mengandung sirkuit dan penambahan satu sisi pada graf akan membuat hanya satu sirkuit.
6. G terhubung dan semua sisinya adalah jembatan.



(a) Pohon

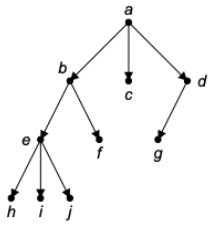


(b) Bukan pohon

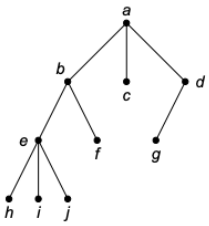
Gambar 3 Pohon dan Bukan Pohon
Sumber: *https://informatika.stei.itb.ac.id/~rinaldi.munir/*

C. Pohon Berakar dan Pohon Biner

Pohon berakar (*rooted tree*) adalah pohon yang salah satu simpulnya diperlakukan sebagai akar dan sisi-sisinya memiliki arah (merupakan graf berarah juga). Karena arahnya sudah jelas, yaitu keluar dari akar, tanda panah pada sisi dapat dihilangkan.



(a) Pohon berakar dengan sisi berpanah



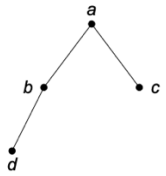
(b) Pohon berakar tanpa sisi berpanah

Gambar 4 Pohon Berakar dengan Sisi Berpanah
Sumber: *https://informatika.stei.itb.ac.id/~rinaldi.munir/*

Berikut adalah beberapa istilah pada pohon berakar.

1. Anak (*child*) dan Orangtua (*parent*)
   Pada Gambar 7, *a* adalah orangtua dari *b, c*, dan *d*. Begitupun sebaliknya, *b, c*, dan *d* adalah anak dari *a*.
2. Derajat (*degree*)
   Derajat suatu simpul adalah banyaknya anak pada simpul tersebut.
3. Daun (*leaf*)
   Daun adalah simpul yang tidak mempunyai anak (atau berderajat nol).

Pohon biner adalah salah satu jenis pohon berakar yang setiap simpulnya memiliki maksimal dua buah anak. Pada umumnya, digunakan terminologi anak kiri (*left child*) dan anak kanan (*right child*) untuk menyebut anak dari suatu simpul pada pohon biner.



Gambar 5 Pohon Biner
Sumber: *https://informatika.stei.itb.ac.id/~rinaldi.munir/*

Dalam pemrograman, struktur data pohon biner dapat diimplementasikan dengan beberapa cara. Implementasi yang lebih umum adalah dengan struktur *node* berkait. Namun, pohon biner juga sebenarnya dapat diimplementasikan dengan struktur data *array*.

D. Bin Packing Problem

Masalah yang dibahas pada permasalahan ini adalah, ”diberikan suatu *bin* berukuran tetap (*fixed size*), bagaimana cara menata balok-balok sehingga *space* pada *bin* dapat digunakan secara optimal?”. Permasalahan ini dikategorikan sebagai *nondeterministic polynomial* (NP) *problem*. Untuk benar-benar mengoptimasi *space* pada *bin* yang digunakan, dibutuhkan algoritma yang kompleks. Oleh karena itu, untuk memecahkan permasalahan ini, algoritma yang pada umumnya digunakan adalah algoritma pendekatan (*approximation algorithms*). Berikut adalah beberapa algoritma pendekatan untuk memecahkan *bin packing problem*.

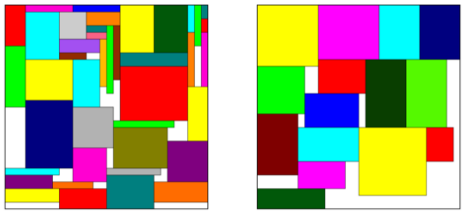
1. Next Fit
   Next Fit adalah algoritma yang naif. Pada algoritma ini, balok-balok disusun langsung secara sekuensial (sesuai urutan masuknya).

2. First Fit
   Pada algoritma ini, balok akan diletakkan pada *space* yang memenuhi syarat yang pertama kali ditemukan.
3. Best Fit
   Pada algoritma ini, balok akan diletakkan pada *space* yang memenuhi syarat yang berukuran paling kecil. Tujuannya adalah agar *space* pada *bin* tidak terpartisi menjadi bagian yang kecil-kecil. Algoritma ini merupakan algoritma yang sangat lambat karena harus mengecek keseluruhan kemungkinan yang ada.
4. First Fit Decreasing
   Persis seperti algoritma First Fit, namun balok-balok yang akan disusun diurutkan mengecil berdasarkan ukurannya.
5. Best Fit Decreasing
   Persis seperti algoritma Best Fit, namun balok-balok yang akan disusun diurutkan mengecil berdasarkan ukurannya.

Berdasarkan referensi [1], secara empiris, urutan algoritma pendekatan dari yang paling optimal:

1. Best Fit Decreasing dan First Fit Decreasing
2. First Fit
3. Next Fit
4. Best Fit

Ada banyak variasi dari permasalahan *bin packing*. Salah satunya adalah *two dimensional* (2D) *bin packing*. Pada 2D *bin packing problem, bin* yang disediakan berupa *space* dua dimensi (memiliki atribut lebar (*width*) dan tinggi (*height*)), dan balok-balok yang akan diletakkan ke dalam *bin* juga merupakan benda dua dimensi.



Gambar 6 2D *Bin Packing*
Sumber: *https://journals.plos.org/plosone/*

Dalam dunia nyata, konsep 2D *bin packing problem* dapat diaplikasikan untuk memecahkan banyak permasalahan. Contohnya, konsep ini dapat diaplikasikan untuk mengoptimalkan *space* yang ada pada bidang:

1. desain *web*
2. pemotongan pada industri kayu, logam, dan kaca
3. *packing* (transportasi dan *warehousing*)

E. Mekanisme Pengurutan pada 2D Bin Packing Problem

Pada bagian sebelumnya, disebutkan bahwa salah satu algoritma pendekatan yang digunakan dalam memecahkan permasalahan *bin packing* adalah First Fit Decreasing dan Best Fit Decreasing. Kedua algoritma ini melibatkan pengurutan balok-balok. Pada permasalahan 2D *bin packing*, setiap objek, baik *bin* maupun *balok* memiliki atribut lebar (*width*) dan tinggi (*height*). Akibatnya, ada banyak versi pengurutan yang dapat digunakan untuk persoalan ini. Balok-balok dapat diurutkan berdasarkan:

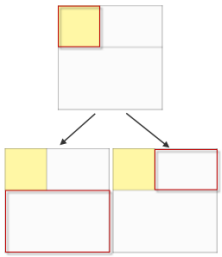
1. lebar balok (*width*)
2. tinggi balok (*height*)
3. luas balok
4. maksimum antara lebar dan tinggi balok (max(*width, height*))

Menurut referensi [2], mekanisme pengurutan yang paling optimal adalah mekanisme pengurutan berdasarkan maksimum antara lebar dan tinggi balok.

III. IMPLEMENTASI

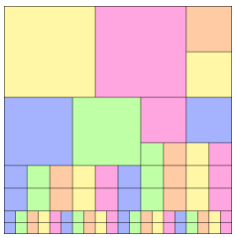
A. Pendekatan Permasalahan dengan Konsep Pohon Biner

Untuk melihat bagaimana permasalahan ini terkait dengan pohon biner, proses peletakan balok perlu dilihat secara visual. Misalkan ada *bin* dua dimensi dengan ukuran tetap dan ada beberapa balok yang akan disusun ke dalam *bin* tersebut. Untuk kemudahan dan kerapihan, balok pertama akan selalu diletakkan di ujung kiri atas. Saat balok pertama diletakkan di sudut atas kiri, dapat terlihat bahwa *bin* yang awalnya kosong ”terpisah” menjadi dua bagian *bin* kosong yang lebih kecil (*space* di kanan dan *space* di bawah).



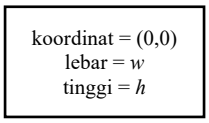
Gambar 7 Peletakan Balok Pertama pada *Bin*
Sumber: *https://codeincomplete.com/articles/bin-packing/*

Di sini, masing-masing *space* kanan dan *space* bawah dapat dianggap sebagai *bin* baru, yaitu *bin* kanan dan *bin* bawah. Balok selanjutnya pun dapat dimasukkan ke *bin* kanan atau kiri, sesuai dengan strategi yang digunakan nantinya. Saat balok kedua dimasukkan ke salah satu di antara *bin* kanan dan *bin* bawah yang baru, *bin* tersebut akan terpecah menjadi dua bagian lagi. Saat memasukkan balok selanjutnya, *bin* akan terus terbagi menjadi dua. Dapat dilihat bahwa struktur *bin* dapat direpresentasikan dengan pohon biner.

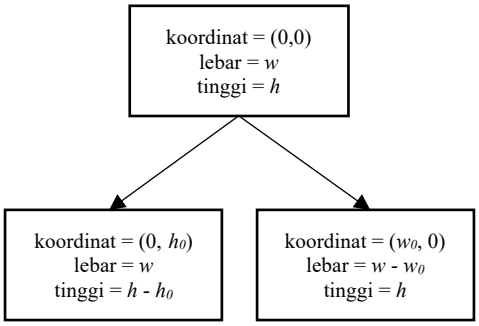


Gambar 8 Peletakan Berbagai Balok pada *Bin*
Sumber: *https://codeincomplete.com/articles/bin-packing/*

Berikut adalah visualisasi peletakan balok dengan pohon biner. *w* adalah lebar *bin* dan *h* adalah tinggi *bin*.



(a) Pohon biner awal



(b) Pohon biner setelah balok dengan lebar w_0 dan tinggi h_0 dimasukkan

Gambar 9 Pohon Biner pada Peletakan Balok pada *Bin*
Sumber: Dokumentasi Penulis

Dari Gambar 9 tersebut, dapat dilihat bahwa peletakan satu balok pada *bin* akan menambah dua simpul daun pada pohon biner (dalam beberapa kasus, jika luas simpul daun tersebut adalah nol, simpul daun tersebut tidak perlu dianggap ada). Balok-balok yang selanjutnya akan dimasukkan hanya bisa diletakkan pada simpul-simpul daun.

B. Penerapan Pendekatan dengan Konsep Pohon Biner

Dalam penerapan ini, digunakan algoritma pendekatan First Fit Decreasing dan Best Fit Decreasing, yang berdasarkan pemaparan sebelumnya merupakan algoritma pendekatan yang paling optimal. Mekanisme pengurutan balok yang digunakan adalah pengurutan berdasarkan maksimum antara lebar dan tinggi balok (max(*width, height*)).

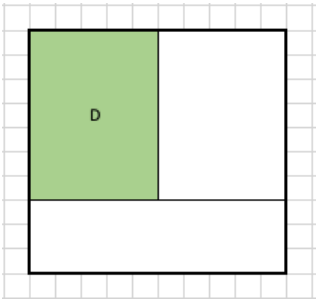
Misalkan ada sebuah *bin* berukuran 10 × 10. Balok A, B, C, dan D, sesuai deskripsi pada Tabel 1, akan diletakkan ke dalam *bin* tersebut sesuai dengan aturan peletakan balok yang dipilih.

Tabel 1 Deskripsi Balok A, B, C, D

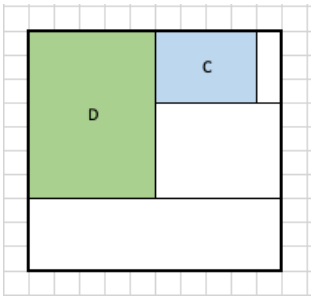
Kode Balok	lebar (w)	tinggi (h)	max(w, h)
A	3	3	3
B	1	2	2
C	4	3	4
D	5	7	7

Sebelum meletakkan balok-balok tersebut, balok-balok tersebut harus diurutkan terlebih dahulu berdasarkan maksimum antara lebar dan tinggi balok. Dapat dilihat bahwa urutan balok dari yang terbesar ke terkecil adalah D, C, A, lalu B.

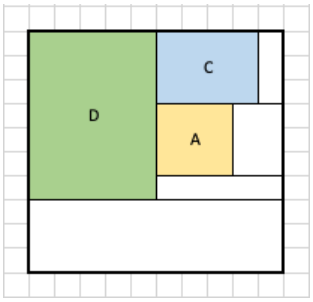
Selanjutnya, balok-balok akan diletakkan pada *bin* sesuai dengan algoritma pendekatan yang dipilih. Jika digunakan algoritma First Fit Decreasing, proses peletakan balok-balok adalah sebagai berikut.



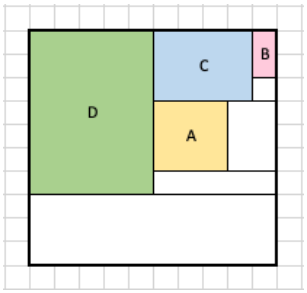
(a) Memasukkan balok D



(b) Memasukkan balok C



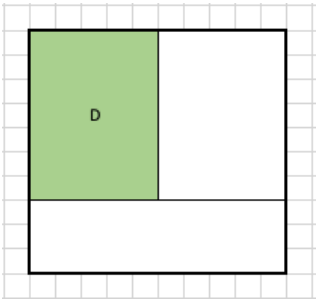
(c) Memasukkan balok A



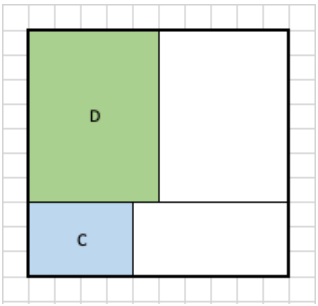
(d) Memasukkan balok B

Gambar 10 Proses Peletakan Balok D, C, A, dan B pada Algoritma First Fit Decreasing
Sumber: Dokumentasi Penulis

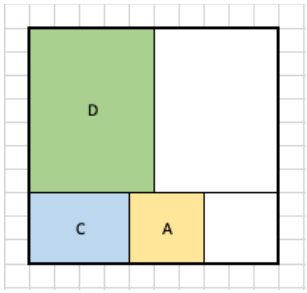
Namun, jika algoritma yang dipilih adalah Best Fit Decreasing, proses peletakan balok-balok adalah sebagai berikut.



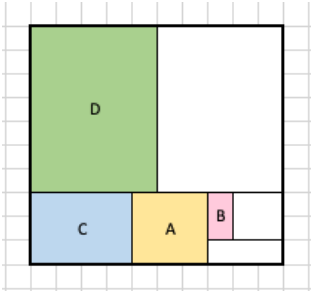
(a) Memasukkan balok D



(b) Memasukkan balok C



(c) Memasukkan balok A



(d) Memasukkan balok B

Gambar 11 Proses Peletakan Balok D, C, A, dan B pada Algoritma Best Fit Decreasing
Sumber: Dokumentasi Penulis

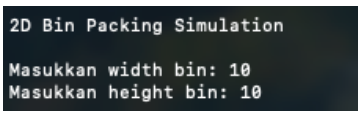
C. Implementasi dalam Program

Untuk memvisualisasikan peletakan balok secara fleksibel dan cepat, penulis juga membuat program simulasi yang ditulis dalam bahasa pemrograman C. Pada program ini, digunakan beberapa struktur data, yaitu sebagai berikut.

1. *Ordered Array*
   Struktur data ini digunakan untuk menyimpan balok-balok yang akan disusun ke dalam *bin*. Struktur data ini dinamai `listBlock` dan terletak pada *abstract data type* (ADT) `listBlock`.
2. Pohon Biner
   Struktur data ini berfungsi untuk mengatur peletakan balok-balok yang ada pada `listBlock` ke dalam *bin*. Pada program ini, pohon biner diimplementasikan dalam bentuk *array*. Struktur data ini dinamai `treeArr` dan terletak pada *abstract data type* (ADT) `treeArr`.
3. Matriks
   Struktur data ini digunakan untuk memvisualisasikan peletakan balok dalam *bin*. Struktur data ini dinamai `matBin` dan terletak pada *abstract data type* (ADT) `matBin`.

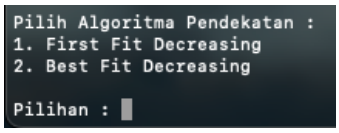
Keseluruhan simulasi 2D *bin packing* dapat dijalankan di program utama (*main*). Di program utama, pengguna akan diminta untuk memasukkan ukuran *bin*, algoritma pendekatan yang digunakan, serta data balok-balok yang akan diletakkan ke dalam *bin* (tidak perlu terurut karena program yang akan mengurutkan). Pengguna juga dapat menghapus balok yang sudah diletakkan ke dalam *bin*.

Sebagai contoh, di sini dipilih *bin* berukuran 10 × 10. Balok A, B, C, dan D, sesuai deskripsi pada Tabel 1, akan diletakkan ke dalam *bin* tersebut.



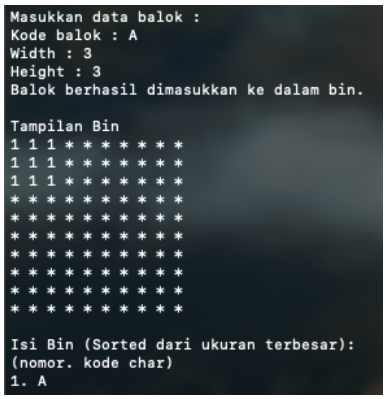
Gambar 12 Memasukkan Ukuran *Bin*
Sumber: Dokumentasi Penulis

Pengguna dapat memilih algoritma yang diinginkan, yaitu antara First Fit Decreasing atau Best Fit Decreasing.

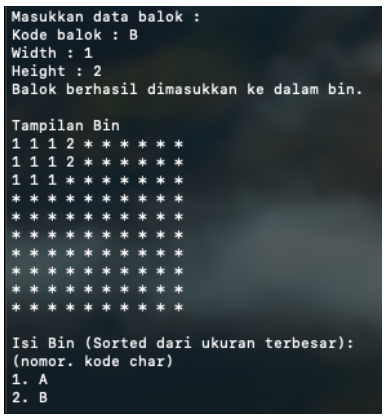


Gambar 13 Memilih Algoritma Pendekatan
Sumber: Dokumentasi Penulis

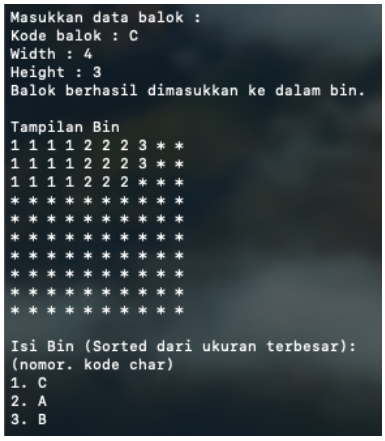
Jika algoritma yang dipilih adalah First Fit Decreasing, proses peletakan balok dari A hingga D adalah sebagai berikut.



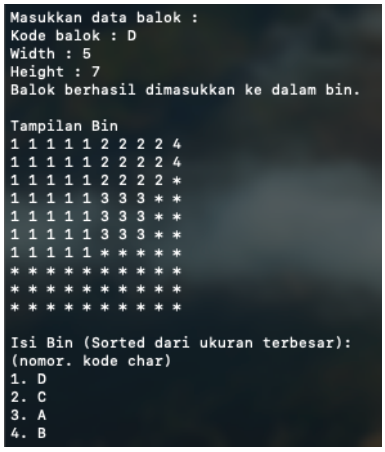
(a) Memasukkan balok A



(b) Memasukkan balok B



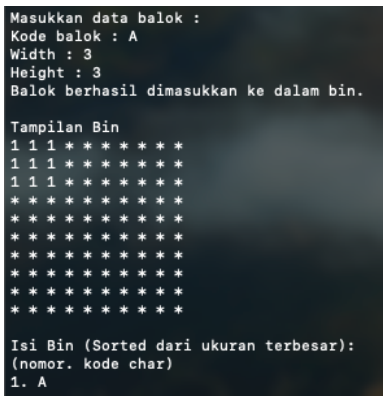
(c) Memasukkan balok C



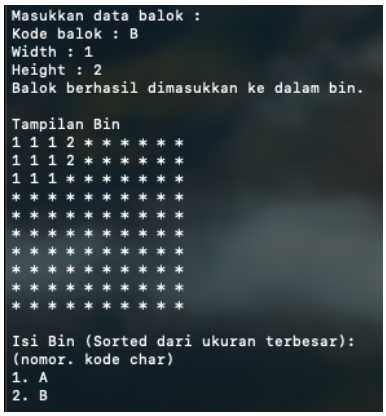
(d) Memasukkan balok D

Gambar 14 Proses Peletakan Balok A, B, C, dan D pada Algoritma First Fit Decreasing pada Program Simulasi
Sumber: Dokumentasi Penulis

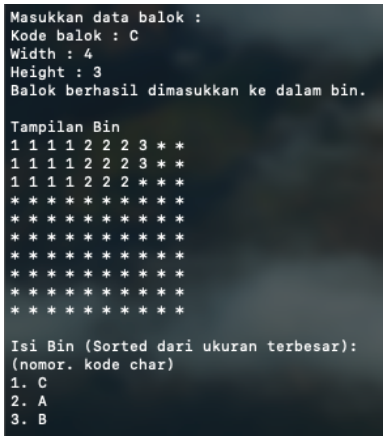
Jika algoritma yang dipilih adalah Best Fit Decreasing, proses peletakan balok dari A hingga D adalah sebagai berikut.



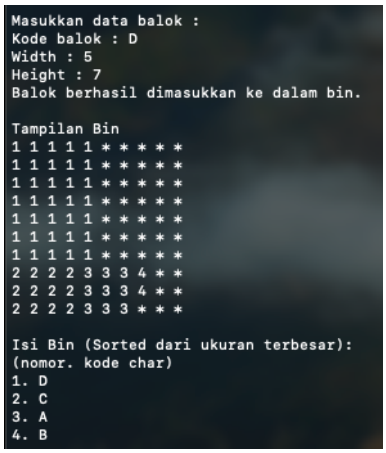
(a) Memasukkan balok A



(b) Memasukkan balok B



(c) Memasukkan balok C



(d) Memasukkan balok D

Gambar 15 Proses Peletakan Balok A, B, C, dan D pada Algoritma Best Fit Decreasing pada Program Simulasi
Sumber: Dokumentasi Penulis

D. Kekurangan Algoritma

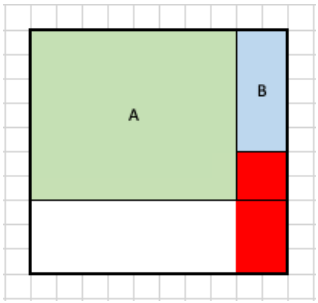
Pada algoritma ini, dapat dilihat bahwa demi keteraturan, suatu balok hanya dapat diletakkan dalam suatu *node bin*. Suatu balok tidak bisa diletakkan pada *space* yang tersusun dari dua atau lebih *node bin* berbeda, meskipun sebenarnya balok tersebut bisa diletakkan pada *space* tersebut.

Misalnya, dimiliki ukuran *bin* sebesar 10 × 10 dan dimiliki data balok sebagai berikut.

Tabel 2 Deskripsi Balok A, B, C

Kode Balok	lebar (w)	tinggi (h)	max(w,h)	Urutan
A	8	7	8	1
B	2	5	5	2
C	1	4	4	3

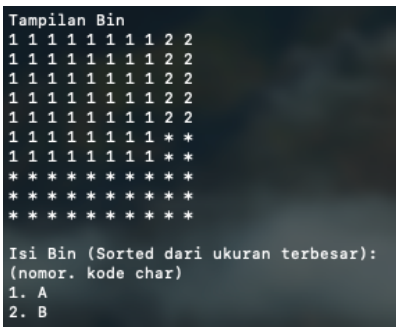
Setelah balok A dan balok B dimasukkan, akan terlihat susunan sesuai pada Gambar 16. Secara kasat mata, seharusnya balok C dapat dimasukkan pada *space* yang kosong yang ditandai oleh warna merah.



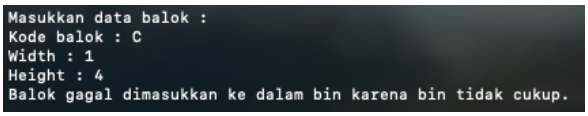
Gambar 16 Susunan Balok A dan B
Sumber: Dokumentasi Penulis

Namun, berdasarkan algoritma yang dipakai, baik dengan strategi First Fit Decreasing maupun Best Fit Decreasing, balok ketiga tidak dapat dimasukkan ke dalam *bin*. Perhatikan *node-node bin* yang *available* pada Gambar 16. Tidak ada *node* yang cukup untuk balok C. Oleh karena itu, balok C tidak akan bisa dimasukkan ke dalam *bin*.

Simulasi pada program juga menunjukkan hasil yang sama, baik dengan mekanisme pengurutan First Fit Decreasing maupun Best Fit Decreasing.



(a) Tampilan setelah memasukkan balok A dan B



(b) Tampilan saat berusaha memasukkan balok C

Gambar 17 Simulasi Kekurangan Algoritma
Sumber: Dokumentasi Penulis

IV. SIMPULAN

Konsep pohon biner dapat digunakan sebagai salah satu pendekatan untuk memecahkan permasalahan 2D *bin packing* dengan algoritma pendekatan First Fit Decreasing maupun Best Fit Decreasing. Namun, pendekatan dengan konsep pohon biner ini juga memiliki kekurangan, yaitu suatu balok tidak bisa diletakkan pada *space* yang tersusun dari dua atau lebih *node bin* berbeda, meskipun sebenarnya balok tersebut bisa diletakkan pada *space* tersebut.

V. UCAPAN TERIMA KASIH

Penulis mengucapkan terima kasih kepada:

1. Tuhan Yang Maha Esa,
2. orang tua penulis yang senantiasa membantu dan menyemangati penulis,
3. Fariska Zakhralativa Ruskanda, S.T., M.T. selaku dosen pengampu mata kuliah IF2120 Matematika Diskrit, dan
4. teman-teman yang sudah mendukung penulis hingga saat ini.

LAMPIRAN

Link repository GitHub program simulasi:
https://github.com/ashnchiquita/2D-Bin-Packing-using-Binary-Tree

REFERENSI

[1] Jones, Eisah, *Bin Packing Algorithms.* 2019. *https://www.eisahjones.com/sorting-algorithms*. Diakses pada 8 Desember 2022.

[2] Kontributor Code In Complete, *Binary Tree Bin Packing Algorithm.* 2011. *https://codeincomplete.com/articles/bin-packing/*. Diakses pada 8 Desember 2022.

[3] Kontributor Geeks For Geeks. *Bin Packing Problem (Minimize number of used Bins).* 2022. *https://www.geeksforgeeks.org/bin-packing-problem-minimize-number-of-used-bins/.* Diakses pada 8 Desember 2022.

[4] Munir, Rinaldi, *IF2120 Matematika Diskrit Semester I Tahun 2022/2023*. 2022. *https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/matdis.htm*. Diakses pada 8 Desember 2022.

[5] Wengrow, Jay, *A Common-Sense Guide to Data Structures and Algorithms*. Raleigh: The Pragmatic Bookshelf, 2020.

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 12 Desember 2022

Chiquita Ahsanunnisa
13521129