Penyelesaian Persoalan *Tiling* dengan Pendekatan *Dynamic Programming*

Leonardus Brandon Luwianto (13519102)
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail (gmail): leonardusluwianto@gmail.com

***Abstrak*—Program dinamis (*dynamic programming*) merupakan metode algoritme yang akhir-akhir ini banyak digunakan karena kemampuannya menyelesaikan berbagai persoalan yang cukup kompleks, misalnya yang berkaitan dengan optimasi (maksimasi atau minimasi) atau persoalan dengan *overlapped subproblems*. Salah satu penerapannya dibahas dalam makalah ini yaitu digunakan untuk menyelesaikan persoalan *tiling* (teselasi). Persoalan dengan *overlapped subproblems* ini diselesaikan menggunakan metode program dinamis agar memperoleh penyelesaian yang lebih mangkus/efektif.**

Kata kunci—tiling, ubin, program, dinamis, rekursif

I. PENDAHULUAN: *TILING PROBLEM*

Bermula pada tahun 1961, seorang matematikawan bernama Hao Wang mengemukakan Wang *tile* (atau Wang *domino*), yaitu sebuah unit *tile* berukuran persegi dengan setiap sisi yang diwarnai. Sebuah set ubin (*tile*) Wang yang terpilih dan duplikatnya disusun bersebelahan dengan warna yang sama, tanpa memutar atau memantulkannya. Wang menduga jika sebuah set ubin Wang disusun pada bidang, akan terbentuk teselasi (*tiling*) yang periodik. Gagasan ubin Wang yang mengharuskan agar ubin dengan sisi berwarna sama saling berbatasan ini mirip dengan permainan domino sehingga *Wang tile problem* disebut juga *Wang domino problem.*

Tiling problem atau yang juga dikenal sebagai *domino problem* merupakan persoalan keputusan yang mempertanyakan apakah sebuah set ubin terbatas T menerima paling sedikit satu ubin valid t: $Z^2 \rightarrow T$. Dengan kata lain, *domino problem* mempermasalahkan apakah ada prosedur efektif yang dapat menyelesaikan masalah untuk semua set domino. Pada tahun 1966, seorang murid dari Wang yang bernama Robert Berger membuktikan *domino problem* bersifat *undecidable*. Hasil tersebut didasarkan pada set *tile* yang tidak periodik (*aperiodic*).

II. PROGRAM DINAMIS (*DYNAMIC PROGRAMMING*)

Program dinamis (*dynamic programming*) merupakan metode pemecahan masalah yang dikembangkan oleh Richard Bellman pada 1950-an dengan cara menguraikan solusi menjadi sekumpulan tahapan (*stage*), sedemikian sehingga solusi persoalan dapat dipandang sebagai rangkaian keputusan yang saling berkaitan. Kata "program" tidak ada kaitannya dengan pemrograman, tetapi mengacu pada "perencanaan (*planning*)".

Program dinamis adalah teknik untuk menyelesaikan masalah dengan upapersoalan yang tumpang tindih *(overlapping subproblems)*. Upapersoalan ini muncul dari rekurens suatu solusi dari persoalan utama menjadi upapersoalan yang lebih kecil. Alih-alih menyelesaikan upapersoalan tumpang tindih secara berulang, program dinamis menyelesaikan setiap upapersoalan hanya sekali dan mencatat hasilnya dalam sebuah tabel hingga solusi persoalan utamanya diperoleh. Ada dua atribut utama yang harus dimiliki masalah agar pemrograman dinamis dapat diterapkan: substruktur yang optimal dan sub-masalah yang tumpang tindih. Jika suatu masalah dapat diselesaikan dengan menggabungkan solusi optimal untuk sub-masalah *tidak tumpang tindih*, strategi yang digunakan adalah algoritme *Divide and Conquer*. Adapun perbedaan algoritme *Greedy* dengan Program dinamis yaitu algoritme *Greedy* hanya satu rangkaian keputusan saja yang dihasilkan sedangkan pada Program dinamis lebih dari satu rangkaian keputusan yang dipertimbangkan.

Program dinamis juga dapat digunakan untuk menyelesaikan persoalan-persoalan optimasi (maksimasi atau minimasi). Pada program dinamis, rangkaian keputusan yang optimal dibuat dengan menggunakan Prinsip Optimalitas. Prinsip Optimalitas yaitu jika solusi total optimal, bagian solusi sampai tahap ke-k juga optimal. Prinsip Optimalitas berarti bahwa jika kita bekerja dari tahap k ke tahap $k + 1$, kita dapat menggunakan hasil optimal dari tahap k tanpa harus kembali ke tahap awal. Terdapat dua pendekatan yang digunakan dalam program dinamis yaitu

1. Program Dinamis Maju (*top-down approach*)
   Perhitungan dilakukan dari tahap 1, 2, …, $n – 1$, n

2. Program Dinamis Mundur (*bottom-up approach*)
   Perhitungan dilakukan dari tahap n, $n – 1$, …, 2, 1

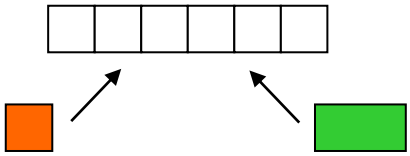
Langkah-langkah pengembangan algoritme program dinamis dapat dirumuskan: 1. Karakteristikkan struktur solusi optimal (menentukan tahap, variable keputusan, status, dsb.); 2. Definisikan secara rekursif nilai solusi optimal (hubungan nilai optimal suatu tahap dengan tahap sebelumnya); 3. Hitung nilai solusi optimal secara maju atau mundur; 4. Rekonstruksi solusi optimal (opsional).

III. Analisis Pemecahan Masalah

A. *Deskripsi Persoalan*

Tentu banyak persoalan *tiling* yang dapat diselesaikan menggunakan program dinamis. Namun, dalam makalah ini akan digunakan suatu persoalan *tiling* yang dapat dijadikan contoh ilustrasi bagaimana langkah-langkah penyelesaiannya menggunakan metode program dinamis. Perhatikan deskripsi persoalannya berikut!

Diberikan sebuah papan dengan panjang 6 satuan, lalu pertanyaannya adalah berapa banyak cara memasang ubin jingga dengan panjang 1 satuan dan ubin hijau dengan panjang 2 satuan pada papan tersebut? Asumsikan jumlah kedua macam ubin tidak terbatas. Untuk lebih jelasnya, Anda dapat melihat ilustrasi gambar berikut.



Gambar 3.1 Ilustrasi persoalan *tiling*

Persoalan ini akan diselesaikan menggunakan metode program dinamis dengan pendekatan *top-down*. Definisikan f(n) sebagai banyak cara memasang ubin pada papan dengan panjang n, sehingga status mula-mula yaitu papan belum dipasangi ubin sama sekali dapat dituliskan dengan f(6) karena panjang papan sebesar 6 satuan. Pada persoalan ini, tahap (*k*) adalah proses memasang ubin pada papan. Status (*n*) menyatakan kapasitas papan yang tersisa setelah memasang ubin pada tahap sebelumnya. Karena hanya ada dua jenis ubin: jingga dan hijau, setiap status dibagi menjadi dua upastatus: papan dipasangi ubin jingga dengan panjang 1 satuan dari tahap sebelumnya dan papan dipasangi ubin hijau dengan panjang 2 satuan dari tahap sebelumnya. Pada gambar 3.2 dapat dilihat perincian segala kemungkinan cara pemasangan ubin pada papan.

B. *Implementasi Rekursif*

Status awal dimulai dari f(6) yaitu ketika papan belum dipasangi ubin. Relasi rekurens f(n – 1) jika dipasangi satu ubin jingga menjadi f(5) sedangkan relasi rekurens f(n – 2) jika dipasangi satu ubin hijau. Rekurens ini akan berhenti Ketika mencapai basis yaitu f(0) yang berarti papan sudah terpasang tepat enam ubin sehingga tidak dapat dipasangi lagi.

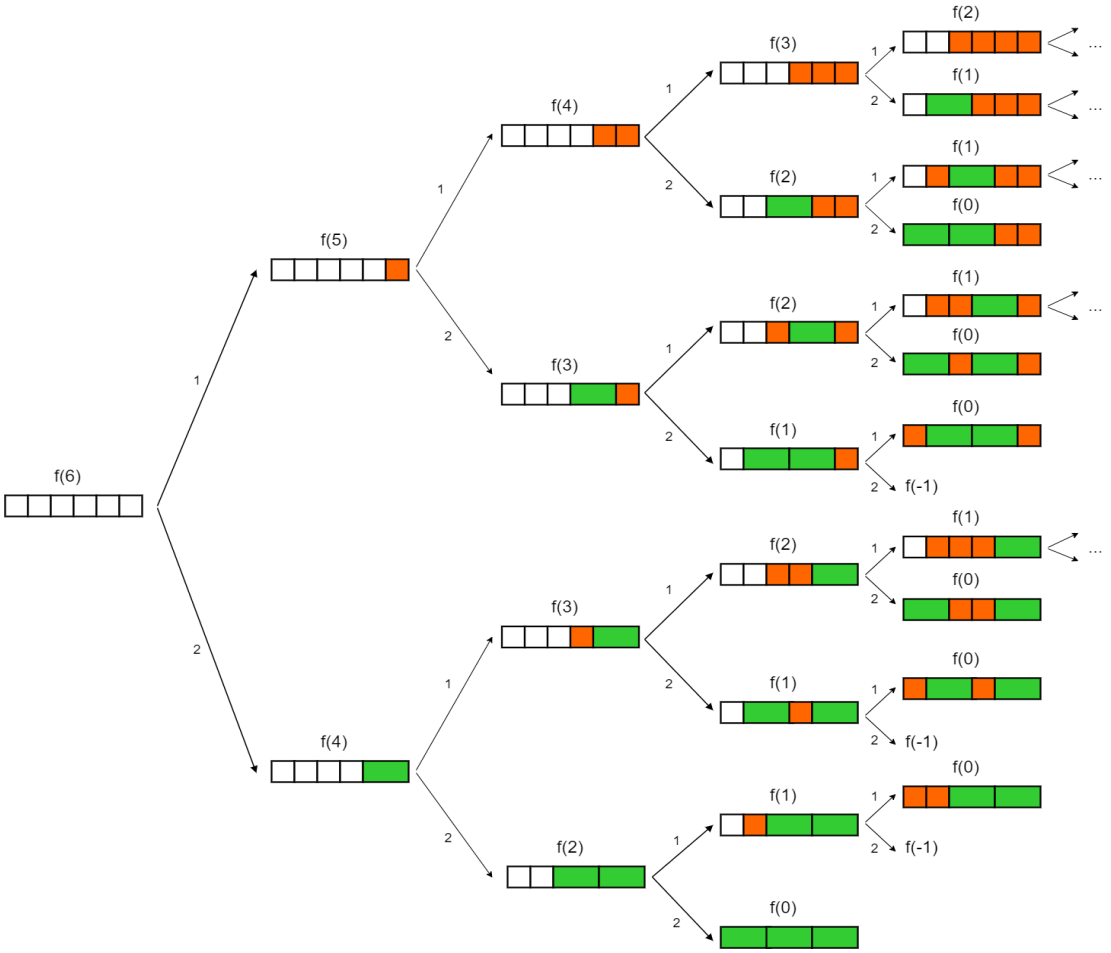
Dari hasil analisis tersebut, kita dapat menentukan hubungan rekursif dari persoalan. Relasi rekurensnya dituliskan sebagai berikut:

$$f(n) = 0, \quad n < 0 \qquad \textit{(basis)}$$

$$f(n) = 1, \quad n = 0 \qquad \textit{(basis)}$$

$$f(n) = f(n-1) + f(n-2), \qquad \textit{(rekurens)}$$

$$n = 1, 2, \ldots$$



Gambar 3.2 Banyak kemungkinan cara pemasangan ubin sesuai persoalan
Sumber: Dokumen penulis

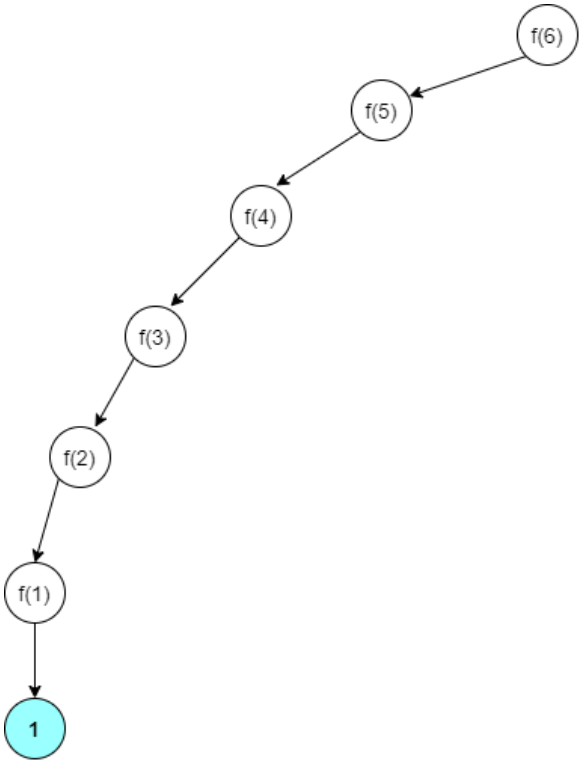
Bayangkanlah relasi rekurens ini sebagai pohon rekursif agar memudahkan pemahaman. Simpul awal f(6) ditambahkan satu ubin jingga, berarti membentuk simpul ekspan f(5) dari relasi rekurens f(n – 1). Karena menerapkan DFS (*Depth First Search*), simpul yang diekspan pertama akan dikunjungi terlebih dahulu. Ekspansi terus berlangsung sampai ketemu simpul f(0). Ketika sampai simpul f(0), kita tahu bahwa papan telah terpasang penuh oleh ubin sehingga kita telah menemukan satu cara memasang ubin pada papan. Oleh karena itu, kita berikan nilai 1 pada simpul daun.

Setelah simpul tidak bisa diekspan lagi, kita *backtrack* ke simpul orang tuanya yaitu f(1). Namun, ubin hijau tidak dapat dipasangkan karena sisa kapasitas papan hanya satu sedangkan ubin hijau panjangnya dua satuan. Jadi, kita berikan nilai 1 pada simpul f(1) tersebut. Kemudian *backtrack* lagi ke simpul oaring tuanya yaitu f(2), ubin hijau dapat dipasangkan karena kapasitas papan tepat tersisa 2 ubin. Jadi, simpul f(2) diekspan menjadi simpul f(0) dari relasi rekurens f(n – 2). Karena sudah mencapai f(0), simpul daun tersebut diberi nilai 1. Simpul f(2) kita berikan nilai 2 karena memuat dua simpul daun bernilai 1 yang sudah kita kunjungi tadi. Dari simpul tersebut, kita *backtrack* lagi dan mengulangi langkah yang sama seperti tadi sampai seluruh simpul telah dikunjungi.

C. Metode Program Dinamis

Namun, cara menghitung banyaknya cara memasang ubin seperti di atas tidak efektif dan menghasilkan jumlah simpul yang eksponensial $O(2^n)$. Tentunya, kita ingin menyelesaikan persoalan tersebut dengan lebih cerdas dan efektif. Oleh karena itu, kita dapat menggunakan metode program dinamis karena ditemukan banyak upapersoalan yang tumpang tindih *(overlapping subproblems)* dan dengan program dinamis, kita dapat menyelesaikan setiap upapersoalan hanya sekali dan mencatat hasilnya dalam sebuah tabel hingga solusi persoalan utamanya diperoleh sehingga kompleksitasnya O(n). Perhatikan langkah-langkahnya sebagai berikut!

Langkah 1

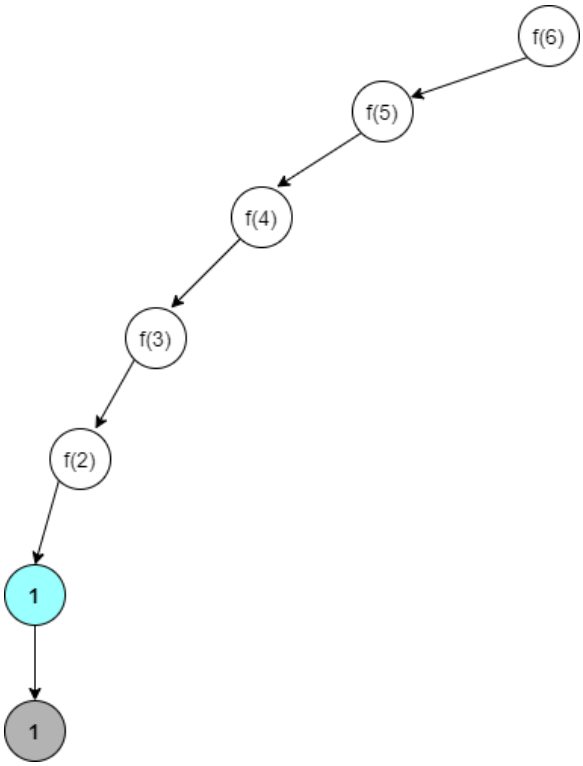


Gambar 3.3 Nilai simpul f(0)
Sumber: Dokumen penulis

f (0)	**1**
f (1)	?
f (2)	?
f (3)	?
f (4)	?
f (5)	?
f (6)	?

Langkah 2

Catat f(0) bernilai 1 yang juga merupakan basis pada tabel untuk kebutuhan yang akan datang. Selanjutnya, ke simpul f(1).

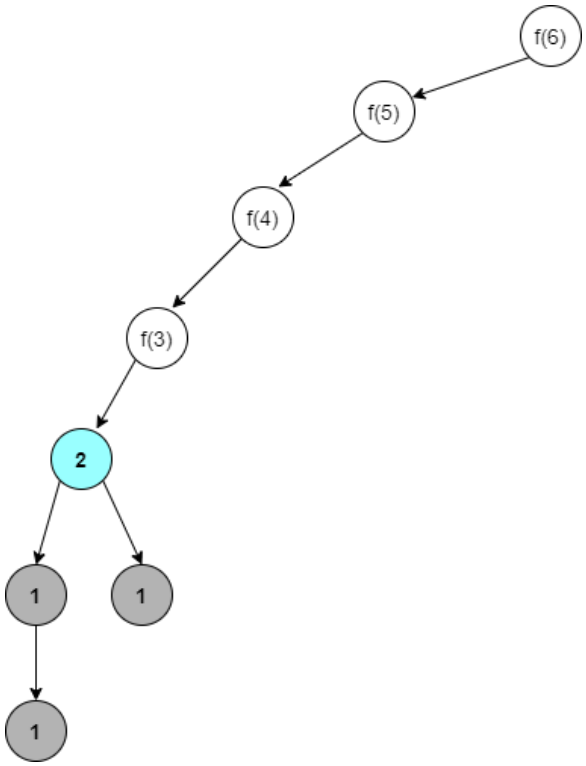


Gambar 3.4 Nilai simpul f(1)
Sumber: Dokumen penulis

f (0)	f (1)	f (2)	f (3)	f (4)	f (5)	f (6)
1	**1**	?	?	?	?	?

Langkah 3

Kemudian *backtrack* ke simpul f(2) dan dapat diekspan menjadi simpul f(0) yang merupakan simpul daun. Karena sudah sudah mencapai basis, berikan nilai 1 sehingga simpul f(2) kita catat nilainya 2.

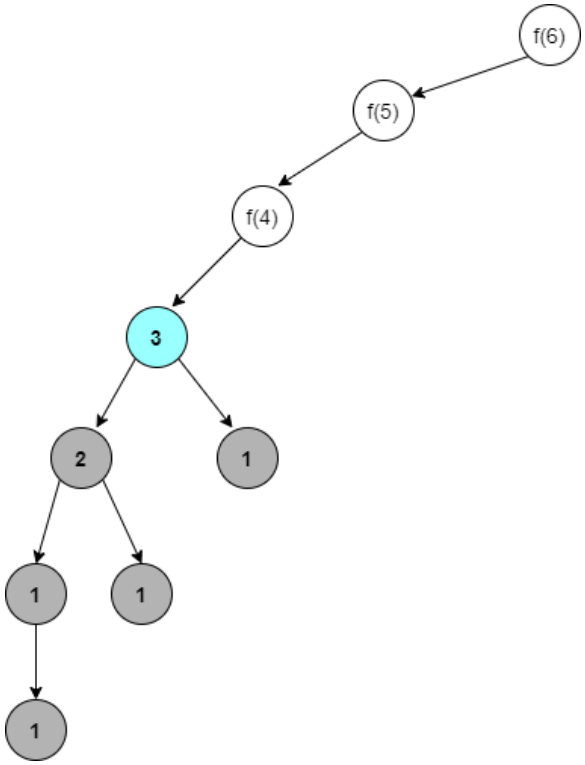


Gambar 3.5 Nilai simpul f(2)
Sumber: Dokumen penulis

f (0)	**1**
f (1)	**1**
f (2)	**2**
f (3)	?
f (4)	?
f (5)	?
f (6)	?

Langkah 4

Kemudian *backtrack* ke simpul f(3) dan dapat diekspan menjadi simpul f(1). Nilai f(1) sudah kita ketahui dari tabel yaitu 1. Jadi, kita catat nilai simpul f(3) adalah 3.

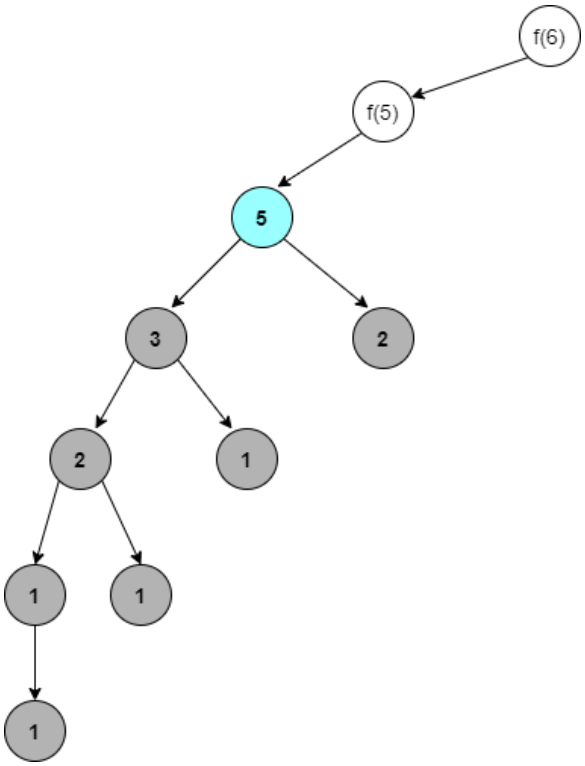


Gambar 3.6 Nilai simpul f(3)
Sumber: Dokumen penulis

f (0)	**1**
f (1)	**1**
f (2)	**2**
f (3)	**3**
f (4)	?
f (5)	?
f (6)	?

Langkah 5

Kemudian *backtrack* ke simpul f(4) dan dapat diekspan menjadi simpul f(2). Nilai f(2) sudah kita ketahui dari tabel yaitu 2. Jadi, kita catat nilai simpul f(4) adalah 5.

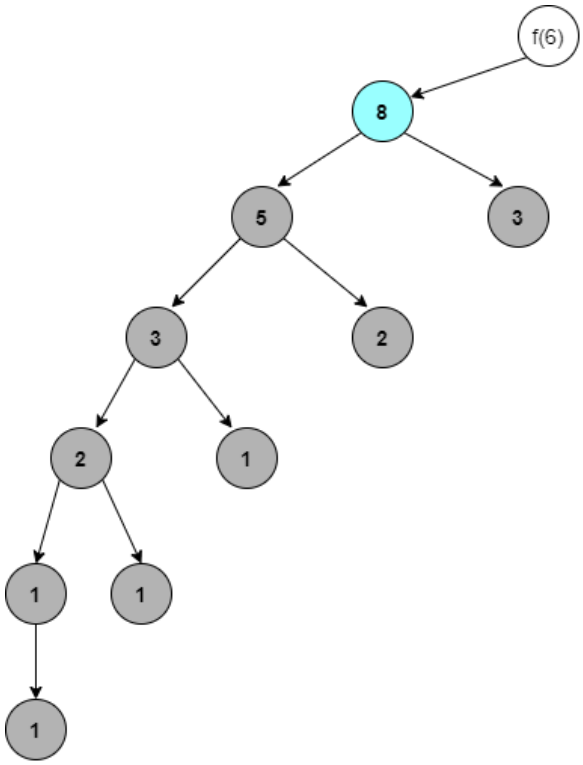


Gambar 3.7 Nilai simpul f(4)
Sumber: Dokumen penulis

f (0)	**1**
f (1)	**1**
f (2)	**2**
f (3)	**3**
f (4)	**5**
f (5)	?
f (6)	?

Langkah 6

Kemudian *backtrack* ke simpul f(5) dan dapat diekspan menjadi simpul f(3). Nilai f(3) sudah kita ketahui dari tabel yaitu 3. Jadi, kita catat nilai simpul f(5) adalah 8.

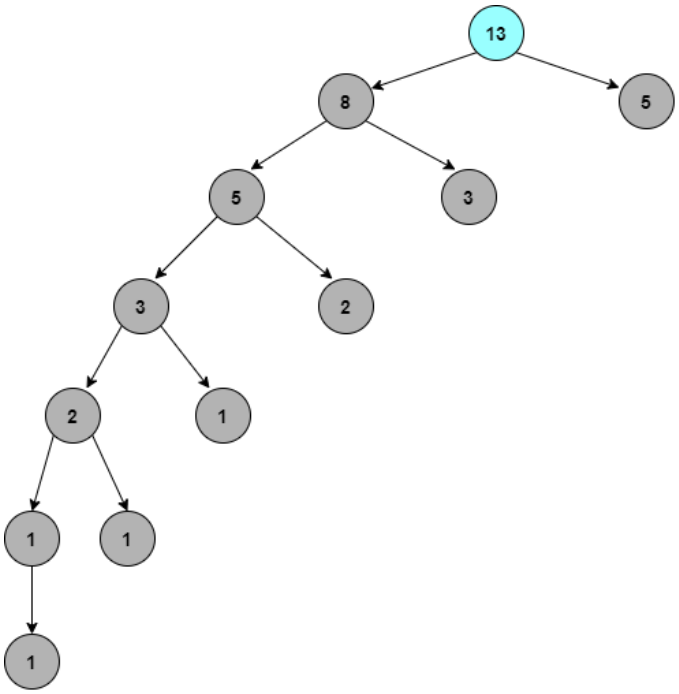


Gambar 3.8 Nilai simpul f(5)
Sumber: Dokumen penulis

f (0)	**1**
f (1)	**1**
f (2)	**2**
f (3)	**3**
f (4)	**5**
f (5)	**8**
f (6)	?

Langkah 7

Kemudian *backtrack* ke simpul f(6) dan dapat diekspan menjadi simpul f(4). Nilai f(4) sudah kita ketahui dari tabel yaitu 5. Jadi, kita catat nilai simpul f(6) adalah 13.



Gambar 3.9 Nilai simpul f(6)
Sumber: Dokumen penulis

f (0)	**1**
f (1)	**1**
f (2)	**2**
f (3)	**3**
f (4)	**5**
f (5)	**8**
f (6)	**13**

Dengan mengetahui nilai simpul f(6), berarti kita telah menemukan solusi persoalan. Jadi, banyaknya cara memasang ubin jingga dengan panjang 1 satuan dan ubin hijau dengan panjang 2 satuan pada papan dengan panjang 6 satuan adalah **13 cara**. Dapat diperhatikan juga bahwa nilai simpul f(0) sampai simpul f(6) membentuk deret Fibonacci.

Demikian langkah-langkah penyelesaian persoalan *tiling* dengan menggunakan pendekatan program dinamis (*dynamic programming*). Dapat disimpulkan bahwa program dinamis merupakan metode yang cocok untuk menyelesaikan persoalan dengan *overlapped subproblems*.

VIDEO LINK AT YOUTUBE

-

UCAPAN TERIMA KASIH

Puji syukur kepada Tuhan Yang Maha Esa atas berkat dan rahmatNya sehingga makalah ini dapat diselesaikan dengan baik dan tepat waktu dalam rangka memenuhi tugas mata kuliah IF2211 – Strategi Algoritme. Penulis mengucapkan terima kasih kepada Ibu Dr. Nur Ulfa Maulidevi, S.T., M.Sc. sebagai dosen pembimbing kelas mata kuliah Strategi Algoritme dan para pihak yang telah menghasilkan karya-karya yang berguna untuk menyelesaikan makalah ini. Penulis berharap agar makalah ini sekiranya dapat berguna bagi pembaca dan masyarakat.

REFERENSI

[1] Kari, J. (2008) On the Undecidability of the Tiling Problem. In: Geffert V., Karhumäki J., Bertoni A., Preneel B., Návrat P., Bieliková M. (eds) SOFSEM 2008: Theory and Practice of Computer Science. SOFSEM 2008. Lecture Notes in Computer Science, vol 4910. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-540-77566-9_7

[2] Levitin, Anany. 2011. *Introduction to the Design & Analysis of Algorithms —3rd ed.* USA.

[3] Munir, Rinaldi. 2020. *Program Dinamis (Dynamic Programming) Bagian* https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Program-Dinamis-2020-Bagian1.pdf (diakses 9 Mei 2021)

[4] https://github.com/williamfiset/algorithms (diakses 9 Mei 2021)

[5] https://projecteuler.net/problem=114 (diakses 9 Mei 2021)

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.

Surakarta, 11 Mei 2021

Leonardus Brandon Luwianto (13519102)