



Dynamic Programming: Studi Kasus OSK/OSP

Muhammad Fairuzi Teguh

CS UI 2017

Pendahuluan

Melalui dokumen ini, kalian akan:

- Menyelesaikan beberapa contoh persoalan DP pada OSK maupun OSP.
- Membiasakan diri untuk "berpikir secara DP".



Contoh Soal 1: Menjelajahi Petak

					N, M
1, 1					

- Diberikan sebuah petak 2 dimensi berukuran $N \times M$.
- Beberapa petak merupakan petak terlarang yang tidak dapat dilalui.
- Kita berada pada petak $(1, 1)$.
- Kita hanya dapat bergerak ke atas dan ke kanan.
- Berapa banyak cara menuju (N, M) ?



Observasi

- Menggunakan kombinatorik membutuhkan inklusi-eksklusi yang cukup banyak.
- Untuk setiap petak (x, y) , ada dua cara untuk menujunya yaitu dari bawah $(x, y - 1)$ atau dari kiri $(x - 1, y)$.
- Karena hanya bisa bergerak ke atas atau kanan, maka dari petak $(x, y - 1)$ kita tidak bisa bergerak ke petak $(x - 1, y)$ begitu pula sebaliknya sehingga kedua kasus tersebut saling lepas.



Formulasi

- Definisikan sebuah fungsi $dp(x, y)$ sebagai banyak cara menuju petak (x, y) jika awalnya kita berada pada $(1, 1)$.
- Kita dapat mencapai petak (x, y) dari petak di bawahnya jika petak (x, y) bukan petak paling bawah ($y > 1$).
- Kita juga dapat mencapai petak (x, y) dari petak di kirinya jika petak (x, y) bukan petak paling kiri ($x > 1$).



Formulasi Rekurens

- Banyak cara menuju petak (x, y) dari petak $(x, y - 1)$ adalah $dp(x, y - 1)$.
- Banyak cara menuju petak (x, y) dari petak $(x - 1, y)$ adalah $dp(x - 1, y)$.
- Karena kedua kasus tersebut saling lepas, banyak cara total adalah penjumlahan dari kedua kasus tersebut.
- Dapat dituliskan:
$$dp(x, y) = dp(x, y - 1) + dp(x - 1, y).$$
- Tentu saja jika petak (x, y) merupakan petak paling kiri maka hanya akan menghitung banyak cara dari petak di bawahnya, begitu pula sebaliknya.



Formulasi Base Case

- Karena pada awalnya berada pada petak $(1, 1)$, maka hanya ada satu cara menuju $(1, 1)$.
- Ini berarti $dp(1, 1) = 1$.
- Kasus ini menjadi *base case*.



Formulasi Petak Terlarang

- Terdapat beberapa petak terlarang yang tidak dapat dilalui.
- Banyak cara menuju petak tersebut tentu 0.
- Ini berarti $dp(x, y) = 0$ jika petak (x, y) merupakan petak terlarang.



Formulasi Akhir

$dp(x, y)$ dapat dirumuskan sebagai berikut:

$$dp(i, c) = \begin{cases} 1, & x = 1 \wedge y = 1 \\ 0, & \text{petak terlarang} \\ dp(x, y - 1), & x = 1 \\ dp(x - 1, y), & y = 1 \\ dp(x, y - 1) + dp(x - 1, y), & x > 1 \wedge y > 1 \end{cases}$$



Mengisi Tabel

Isi petak-petak yang sudah jelas nilainya.

			0		
	0				
				0	
1					

Mengisi Tabel (lanj.)

Isi petak-petak yang berada di sekeliling petak-petak yang sudah terisi.

			0		
	0				
1	2			0	
1	1				

Mengisi Tabel (lanj.)

Lakukan seterusnya hingga seluruh petak terisi.

1	2	6	6	13	28
1	1	4	0	7	15
1	0	3	7	7	8
1	2	3	4	0	1
1	1	1	1	1	1

Jawaban dari soal ini adalah 28.








Contoh Soal 2: Menghitung Kelinci

- Pada awalnya (tahun ke-0), ada 1 kelinci di dalam kandang.
- Seekor kelinci akan melahirkan seekor kelinci setiap tahunnya jika mereka setidaknya berumur 2 tahun.
- Jika diasumsikan mereka tidak pernah mati, berapa kelinci yang ada pada tahun ke-8?



Observasi

Kita dapat membuat tabel untuk mempermudah membuat rumus rekurens.

0	
1	
2	 
3	  

- Pada tahun ke-0, hanya ada kelinci pertama.
- Pada tahun ke-1, kelinci pertama belum dapat melahirkan.
- Pada tahun ke-2, kelinci pertama melahirkan kelinci kedua.
- Pada tahun ke-3, kelinci pertama melahirkan kelinci ketiga.

Observasi (lanj.)

2	 
3	  
4	    
5	       





















- Pada tahun ke-4, kelinci kedua sudah dapat melahirkan.
- Kelinci pertama melahirkan kelinci keempat dan kelinci kedua melahirkan kelinci kelima.
- Pada tahun ke-5, kelinci pertama, kedua, dan ketiga dapat melahirkan berturut-turut kelinci keenam, ketujuh, dan kedelapan.

Formulasi

- Definisikan $f(n)$ sebagai banyak kelinci pada tahun ke- n .
- Banyak kelinci pada tahun ke- n merupakan banyak kelinci yang ada pada tahun sebelumnya ditambah dengan kelinci yang baru saja dilahirkan.



Formulasi Rekurens

0	
1	
2	 
3	  
4	    
5	       

- Perhatikan bahwa kelinci yang dilahirkan pada tahun ke- n pasti dilahirkan oleh kelinci yang ada pada tahun ke- $(n-2)$.
- Banyak kelinci yang baru dilahirkan pada tahun ke- n adalah $f(n-2)$.



Formulasi Rekurens (lanj.)

- Banyak kelinci pada tahun ke- $(n-1)$ adalah $f(n-1)$.
- Karena banyak kelinci tahun ke- $(n-1)$ adalah $f(n-1)$ dan banyak kelinci yang dilahirkan pada tahun ke- n adalah $f(n-2)$ didapat
$$f(n) = f(n-1) + f(n-2)$$



Formulasi Base Case

- Karena kita membutuhkan $f(n - 1)$ dan $f(n - 2)$ untuk menghitung $f(n)$, kita membutuhkan setidaknya 2 *base case*.
- Dari soal didapat $f(0) = 1$.
- Dari tabel didapat $f(1) = 1$.
- Dua kasus ini menjadi *base case*.



Mengisi Tabel

Isi nilai *base case* terlebih dahulu.

n	0	1	2	3	4	5	6	7	8
f(n)	1	1							



Mengisi Tabel (lanj.)

Lanjutkan dengan nilai selanjutnya.

n	0	1	2	3	4	5	6	7	8
f(n)	1	1	2						



Mengisi Tabel (lanj.)

Lakukan hingga selesai.

n	0	1	2	3	4	5	6	7	8
f(n)	1	1	2	3	5	8	13	21	34

Jawaban dari soal ini adalah 34.



Contoh Soal 3: Mengambil Batu

- Pada suatu permainan, terdapat 15 batu dan 2 orang pemain.
- Setiap pemain mendapat giliran mengambil batu bergiliran.
- Pada setiap giliran, seorang pemain dapat mengambil 1, 2, atau 4 batu.
- Tentu saja seorang pemain dapat mengambil n batu jika masih ada setidaknya n batu.
- Seseorang yang mendapatkan giliran saat batu sudah habis dinyatakan kalah.
- Jika mereka berdua bermain optimal, apakah pemain yang mendapat giliran pertama akan memenangkan permainan tersebut?



Observasi

- Kita dapat menggunakan dp untuk permasalahan ini.
- Pada setiap langkah, terdapat 3 kemungkinan yaitu mengambil 1, 2, atau 4 batu.
- Kita ingin mengambil batu sedemikian hingga lawan kita pasti kalah.
- Sebagai contoh, jika ada 2 batu, kita dapat mengambil 1 maupun 2 batu.
- Kita pasti mengambil 2 batu karena lawan kita pasti kalah.



Formulasi

- Definisikan $f(n)$ sebagai kebenaran dari pernyataan "Orang yang mendapat giliran saat ada n batu bisa menang".
- Kemungkinan nilai dari dp hanya dua, apakah *true* (1) atau *false* (0)
- Karena kedua pemain bermain optimal, definisi dp dapat dibuat menjadi "Orang yang mendapat giliran saat ada n batu **pasti** menang"
- Jawaban dari soal ada pada $f(15)$.



Formulasi Rekurens

- Pada setiap langkah, kita dapat mengambil 1, 2, atau 4 batu yang berturut-turut menghasilkan $f(n-1)$, $f(n-2)$, dan $f(n-4)$.
- Kita ingin mengambil batu sehingga sisa batu dapat membuat lawan kalah.
- Ini berarti kita ingin mencari di antara $f(n-1)$, $f(n-2)$, dan $f(n-4)$ yang bernilai *false* (0).
- Sebagai contoh, jika $f(n-1) = 0$, $f(n-2) = 1$ dan $f(n-4) = 0$ maka kita pasti mengambil 1 atau 4 batu.



Formulasi Rekurens (lanj.)

- Jika $f(n-1)$, $f(n-2)$, dan $f(n-4)$ bernilai *true* (1), maka tidak ada langkah yang dapat membuat kita menang.
- Jika ada setidaknya satu yang bernilai *false* (0), maka kita pasti menang.
- Secara formal, $f(n)$ dapat ditulis sebagai $f(n) = \text{not } (f(n-1) \text{ and } f(n-2) \text{ and } f(n-4))$.



Formulasi Base Case

- Jika sudah tidak ada batu, maka yang mendapat giliran kalah.
- Berarti $f(0) = 0$.
- Kasus ini menjadi *base case*.



Mengisi Tabel

Isi base case terlebih dahulu.

n	0	1	2	3	4	5	6	7
f(n)	0							

Mengisi Tabel (lanj.)

Pada $n = 1$, kita hanya dapat mengambil 1 batu yang menghasilkan $f(0)$ yang bernilai 0 sehingga $f(1) = 1$.

n	0	1	2	3	4	5	6	7
f(n)	0	1						



Mengisi Tabel (lanj.)

Pada $n = 2$, kita dapat mengambil 1 atau 2 batu yang menghasilkan berturut-turut $f(1)$ dan $f(0)$. Karena $f(0) = 0$ maka $f(2) = 1$.

n	0	1	2	3	4	5	6	7
f(n)	0	1	1					



Mengisi Tabel (lanj.)

Pada $n = 3$, kita dapat mengambil 1 atau 2 batu yang menghasilkan berturut-turut $f(2)$ dan $f(1)$. Karena tidak ada yang bernilai 0, maka $f(3) = 0$.

n	0	1	2	3	4	5	6	7
f(n)	0	1	1	0				



Mengisi Tabel (lanj.)

Pada $n = 4$, kita dapat mengambil 1, 2, atau 4 batu yang menghasilkan berturut-turut $f(3)$, $f(2)$, dan $f(0)$. Karena $f(3)$ dan $f(0)$ bernilai 0, maka $f(4) = 1$.

n	0	1	2	3	4	5	6	7
f(n)	0	1	1	0	1			



Mengisi Tabel (lanj.)

Lanjutkan hingga selesai.

n	0	1	2	3	4	5	6	7
f(n)	0	1	1	0	1	1	0	1

n	8	9	10	11	12	13	14	15
f(n)	1	0	1	1	0	1	1	0

Jawaban dari soal ini adalah tidak, karena pemain yang mendapat giliran kedua yang akan menang.



Contoh Soal 4: String Biner

- String biner merupakan string yang tersusun atas '0' maupun '1'.
- Kita ingin membuat string biner dengan panjang 6.
- Kita tidak ingin ada substring '100' dalam string tersebut.
- Berapa banyak string biner yang dapat dibuat?



Observasi

0/1	0/1	0/1	0/1	0/1	0/1
-----	-----	-----	-----	-----	-----

- Kita dapat memandang string dengan panjang 6 sebagai 6 kotak yang dapat kita isi dengan 0 atau 1.
- Kita dapat mengisi kotak tersebut dari depan, dengan menjaga 2 kotak sebelumnya dengan kotak yang sedang kita isi tidak menghasilkan substring '100'.

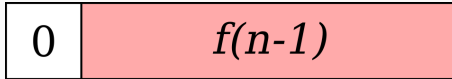


Formulasi $f(n)$

- Definisikan sebuah fungsi $f(n)$ sebagai banyak cara menyusun string biner dengan panjang n tanpa substring '100'.
- Kita dapat mereduksi $f(n)$ menjadi $f(n - 1)$ dengan cara mengisi kotak terdepan.



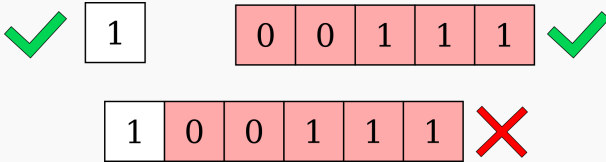
Formulasi Rekurens $f(n)$



- Jika kita isi kotak terdepan dengan '0', maka kita bebas mengisi kotak yang lain asalkan tidak membentuk 100.
- Banyak cara mengisi sisanya adalah $f(n - 1)$.



Formulasi Rekurens $f(n)$ (lanj.)



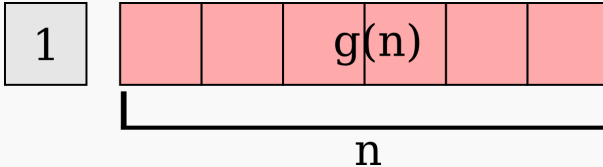
- Jika kita isi kotak terdepan dengan '1', kita **tidak** bebas mengisi $n - 1$ sisanya walaupun dalam $n - 1$ tersebut tidak ada substring 100.
- Banyak cara mengisi sisanya bukan $f(n - 1)$.
- Anggap banyak cara mengisi $n - 1$ sisanya adalah $g(n - 1)$.

Formulasi Rekurens $f(n)$ (lanj.)

- Ada dua kasus pada pengisian kotak pertama.
- Pertama, kita mengisinya dengan '1' dan menghasilkan $f(n - 1)$ kombinasi.
- Kedua, kita mengisinya dengan '0' dan menghasilkan $g(n - 1)$ kombinasi.
- Karena mengisi kotak terdepan dengan 0 atau 1 adalah kasus saling lepas, maka didapat
$$f(n) = f(n - 1) + g(n - 1)$$



Formulasi $g(n)$



- Definisikan sebuah fungsi baru, $g(n)$, sebagai banyak cara menyusun string biner dengan panjang n tanpa substring '100' walaupun digabungkan dengan '1'.

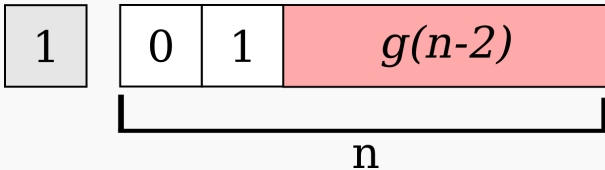
Formulasi Rekurens $g(n)$



- Jika kita isi kotak terdepan dengan '1', maka kita dapat mengisi $n-1$ sisanya dengan syarat tidak ada substring '100' dan jika digabungkan dengan '1' tidak menghasilkan '100'.
- Karena kotak pertama '1', maka kita tidak dapat menghitungnya dengan $f(n-1)$.
- Banyak cara mengisi sisanya adalah $g(n-1)$.



Formulasi Rekurens $g(n)$ (lanj.)



- Jika kita isi kotak terdepan dengan '0', maka kita tidak mungkin mengisi '0' pada kotak selanjutnya karena akan menghasilkan '100'.
- Kita pasti akan mengisi '1' pada kotak selanjutnya.
- Ada $n-2$ kotak yang tersisa.
- Karena bersebelahan dengan kotak '1', maka banyak cara mengisi $n-2$ kotak tersebut adalah $g(n-2)$.



Formulasi Rekurens $g(n)$ (lanj.)

- Ada dua kasus pada saat mengisi kotak terdepan.
- Jika kita mengisinya dengan '1', akan ada $g(n - 1)$ kombinasi yang dihasilkan.
- Jika kita mengisinya dengan '0', akan ada $g(n - 2)$ kombinasi yang dihasilkan.
- Total kombinasi yang ada adalah penjumlahan dari kedua kasus tersebut.
- Dapat dirumuskan:
$$g(n) = g(n - 1) + g(n - 2)$$

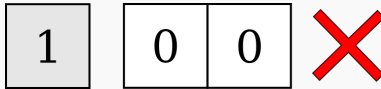


Formulasi Base Case $f(n)$

- Pada $n < 3$, tidak akan ada kombinasi yang membentuk '100'.
- Pada $n = 1$ ada $2^1 = 2$ kombinasi yang mungkin.
- Pada $n = 2$ ada $2^2 = 4$ kombinasi yang mungkin.
- Berarti $f(1) = 2$ dan $f(2) = 4$.



Formulasi Base Case $g(n)$



- Sama seperti $f(n)$, pada $n = 1$ ada 2 kombinasi yang mungkin.
- Pada $n = 2$ ada satu kombinasi yang dapat membuat '100', yaitu '00'.
- Sehingga pada $n = 2$ ada $4 - 1 = 3$ kombinasi yang mungkin.
- Berarti $g(1) = 2$ dan $g(2) = 3$.

Mengisi Tabel

Isi *base case* terlebih dahulu.

n	1	2	3	4	5	6
f(n)	2	4				
g(n)	2	3				

Mengisi Tabel (lanj.)

Isi pada n selanjutnya sesuai rumus $f(n)$ dan $g(n)$.

n	1	2	3	4	5	6
$f(n)$	2	4	7			
$g(n)$	2	3	5			

Mengisi Tabel (lanj.)

Lanjutkan hingga selesai.

n	1	2	3	4	5	6
f(n)	2	4	7	12	20	33
g(n)	2	3	5	8	13	21

Jawaban dari soal ini adalah 33.



Solusi Lain

- Terkadang, ada solusi lain yang dapat digunakan pada permasalahan string biner.
- Solusi tersebut mungkin lebih mudah namun tidak dapat digunakan pada setiap kasus.



Rekurens dengan Pengurangan

- Pada contoh kasus ini, substring yang tidak diperbolehkan adalah '100'.
- Mari mencoba sudut pandang baru terhadap masalah ini.



Formulasi Rekurens

- Untuk mengisi kotak terdepan, terdapat dua cara yaitu dengan '0' atau '1'.
- Banyak cara untuk mengisi sisanya **jika tidak ada larangan** adalah $f(n - 1)$ pada masing-masing kasus.
- Karena ada dua kasus, maka banyak cara total adalah $2 \times f(n - 1)$.



Formulasi Rekurens (lanj.)

- Karena substring '100' dilarang, kita harus menghilangkan kasus tersebut pada perhitungan awal kita.
- Jika ada substring '100' pada tiga kotak terdepan, maka tersisa $n - 3$ kotak.
- Banyak cara mengisi sisanya adalah $f(n - 3)$.
- Banyak cara membuat string biner tanpa substring '100' dapat diperoleh dari banyak cara membuat string biner secara bebas dikurang yang mengandung substring '100'.
- Dapat dituliskan
$$f(n) = 2 \times f(n - 1) - f(n - 3).$$



Formulasi Base Case

- $f(n)$ didefinisikan sebagai banyak string biner dengan panjang n tanpa substring '100'.
- Dengan definisi ini, string dengan panjang kurang dari 3 pasti memenuhi karena tidak mungkin membentuk '100'.
- Berarti $f(1) = 2$ dan $f(2) = 4$.
- Hanya ada satu string dengan panjang 3 yang tidak valid, yaitu '100' itu sendiri.
- Berarti $f(3) = 8 - 1 = 7$.




Mengisi Tabel

n	1	2	3	4	5	6
f(n)	2	4	7	12	20	33


Dapat dilihat hasil sama seperti perhitungan sebelumnya.

Keterbatasan

1	1	1	0	0	0
---	---	---	---	---	---



1	0	1	0	1	0
---	---	---	---	---	---



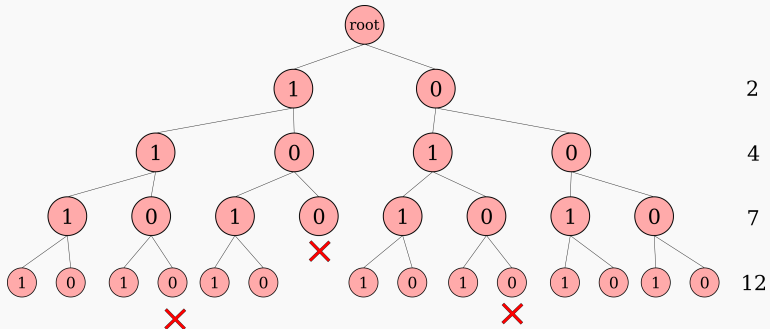
- Cara ini hanya dapat digunakan jika substring yang dilarang tidak dapat dihasilkan dari penggabungan substring tersebut dengan suatu substring valid.
- Sebagai contoh, cara ini tidak dapat digunakan jika substring yang dilarang adalah '11' atau '101'.

Permodelan dengan Pohon

- Untuk mendapat rumus rekurens, terkadang kita dapat memodelkannya dengan pohon (*tree*).
- Jaga agar substring terlarang tidak terbentuk ketika membuat pohon.
- Kita dapat melihat suatu rumus rekurens dari pola yang dihasilkan oleh pohon.
- Untuk melihat pola, buat pohon dengan tinggi yang cukup.



Permodelan dengan Pohon (lanj.)



- Dapat dilihat bahwa:
 - $7 = 4 + 2 + 1$
 - $12 = 7 + 4 + 1$
- Dapat ditebak $f(n) = f(n-1) + f(n-2) + 1$.



Keterbatasan

- Tentu saja kita tidak bisa menggunakan cara ini jika tidak terdapat pola yang jelas.
- Jika kita tidak membuat pohon yang cukup besar, pola bisa jadi salah.
- Terkadang, dengan n yang kecil saja pohon yang dihasilkan sudah terlalu besar.



Kesimpulan

- Dari tiga cara sebelumnya, didapat tiga formulasi rekurens berbeda yaitu
 - $f(n) = f(n-1) + g(n-1)$
 $g(n) = g(n-1) + g(n-2)$
 - $f(n) = 2 \times f(n-1) - f(n-3)$
 - $f(n) = f(n-1) + f(n-2) + 1$
- Ketiga formula tersebut menghasilkan hasil yang sama jika menggunakan *base case* yang tepat.



Latihan Soal 1



Seekor kodok berada pada batu (ditandai dengan warna abu-abu) di ujung kiri dan ingin pergi ke batu yang berada pada ujung kanan. Sayangnya, terdapat daun (ditandai dengan warna hijau) yang tidak dapat dijadikan tempat mendarat (tetapi dapat dia loncati). Dia tidak ingin membuang waktu sehingga hanya akan loncat ke kanan. Jika dengan sekali lompatan dia bisa sampai maksimal pada tiga daun di depannya, berapa banyak cara yang dapat dia lakukan untuk sampai di ujung kanan?



Latihan Soal 2

					s
m					

Seorang atlet berada pada kotak mulai (m) dan ingin pergi ke kotak selesai (s). Dia bisa pergi ke kotak di atasnya maupun di kanannya. Dia dapat melompati sebuah kotak sehingga mencapai dua kotak di depannya (atas maupun kanan). Berapa banyak cara dia untuk sampai di s jika tidak boleh melewati kotak hitam?

Latihan Soal 3

Pada mulanya (menit ke 0), terdapat 2 bakteri di dalam toples. Suatu bakteri akan menghasilkan bakteri baru setiap menitnya jika umur bakteri tersebut setidaknya tiga menit. Pada menit ke berapa sehingga bakteri dalam toples itu setidaknya berjumlah 200?



Latihan Soal 4

Pak Dengklek dan pak Ganesh bermain ambil kelereng yang dilakukan secara bergantian. Seseorang dapat mengambil 1, 3, atau 4 kelereng pada gilirannya. Pemain yang **menghabiskan kelereng yang tersedia** dianggap kalah. Mereka berdua sangat pintar sehingga pasti bermain optimal. Jika pak Dengklek mendapat giliran pertama dan banyak kelereng mula-mula 710, siapakah yang akan menang?



Latihan Soal 5

Pak Dengklek ingin memasang ubin berukuran 1×1 pada lantai berukuran 1×8 (pak Dengklek memandangnya dari kiri ke kanan). Ada dua warna yang tersedia, biru dan merah. Berapa banyak cara pak Dengklek memasang ubin jika:

- dia tidak ingin ada dua ubin merah yang bersebelahan.
- dia ingin ada dua ubin biru yang bersebelahan
- dia tidak ingin ada ubin merah tepat di sebelah kiri ubin biru.
- dia tidak ingin ada pemasangan ubin merah-biru-merah.



Latihan Soal 6

Berapa banyak ternary string (string yang masing-masing karakternya '0', '1', atau '2') dengan panjang 7 yang tidak memiliki substring '012'?



Solusi

1. Suatu kotak berhubungan dengan tiga kotak di kirinya. 8.
2. Suatu kotak berhubungan dengan dua kotak di bawah dan kirinya. 75.
3. Pada menit ke 13 banyak bakteri 202. 13.
4. Pola $7n+1$ dan $7n+3$ kalah. Pak Ganesh.
5. Misalkan ubin merah 0 dan ubin biru 1.
 - Rekurens penjumlahan atau pohon. 55.
 - Inklusi-eksklusi. $2^8 - 55$. 201.
 - Semua cara dapat digunakan. 9.
 - $f(n) = f(n-1) + g(n-1)$
 $g(n) = g(n-1) + f(n-2)$
114.
6. Rekurens pengurangan. 1791.

