Lista de soluções 4

Observação: As soluções são dados em forma de recorrência com uma breve explicação. As técnicas vistas em aula para implementar as recorrências eficientemente e para reduzir a memoria usada podem ser aplicadas para obter melhores soluções.

Exercício 1
Seja $S(i,v) \in \{v, f\}$ a resposta se existe um subconjunto de $[i]$ com soma v. Os valores S satisfazem

$$S(i,v) = \begin{cases} S(i-1,v) \vee S(i-1,v-a_i) & \text{caso } i \geq 1 \text{ e } v \geq a_i \\ S(i-1,v) & \text{caso } i \geq 1 \text{ e } v < a_i \\ v & \text{caso } i = 0 \end{cases}.$$

(O primeiro caso afirma que existe uma solução para o conjunto $[i]$ caso existe uma para o conjunto $[i-1]$, ou nos podemos selecionar i, e existe uma seleção com valor $v - a_i$ entre $[i-1]$. O segundo caso se aplica, se o item i possui valor maior que v e logo não pode ser selecionado, e o terceiro caso é o caso base.)
Logo temos um algoritmo com espaço e tempo $O(nt)$ para calcular a resposta $S(n,t)$.

Exercício 2
Com $M(n,k)$ o número máximo de testes para n "rungs" e k jarras temos a recorrência

$$M(n,k) = \begin{cases} \min_{1 \leq i \leq n} 1 + \max\{M(i-1,k-1), M(n-i,k)\} & \text{caso } k > 1,\ n > 0 \\ n & \text{caso } k = 1 \text{ e } n > 0 \\ 0 & \text{caso } n = 0 \end{cases}.$$

(O primeiro caso afirma que podemos escolher o "rung" que gera o menor número de testes no caso pessimista, considerado o máximo entre os dois casos "quebra" $M(i-1,k-1)$ ou "não quebra" $M(n-i,k)$.)
Isso resolve o problema em tempo e espaço $O(nk)$. Para $n = 500$, por exemplo, a seguinte tabela mostra o menor número de testes um função de k:

$k =$	1	2	3	4	5	6	7	8	
	500	32	15	11	10	10	9	9	...

A partir de sete jarras precisamos nove testes, igual a uma busca binária, e esse número é o menor possível.

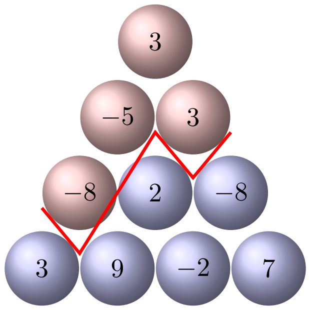
Exercício 3
Seja $S(i,v)$ o menor peso total de um subconjunto do conjunto de itens $[i]$ com valor total v, que cabe na mochila. Os valores S satisfazem

$$S(i,v) = \begin{cases} \min\{S(i-1,v), S(i-1,v-v_i) + w_i\} & \text{caso } i > 0 \text{ e } v_i \leq v \\ S(i-1,v) & \text{caso } i > 0 \text{ e } v_i > v \\ \infty & \text{caso } i = 0 \text{ e } v > 0 \\ 0 & \text{caso } i = 0 \text{ e } v = 0 \end{cases}.$$

O valor da solução ótima do problema da mochila é o maior valor v tal que $S(n,v) \leq W$. O custo da solução via programação dinâmica é $O(nV)$ espaço e tempo.

Exercício 4

Cada seleção de bolas pode ser descrita por uma linha "zig-zag" abaixo deles. Aqui é um exemplo:



Usaremos programação dinâmica para enumerar todas linhas "zig zag". Seja N a altura do triângulo e (d, h) a coordenada de uma bola no triângulo, com $d \in [N]$ sendo o número da diagonal (principal) da bola, e $1 \leq h \leq N - d + 1$ a sua "altura" nessa diagnoal (por exemplo a bola com valor -8 tem coordenadas $(3, 2)$; ver também a tabela abaixo). Vamos escrever a_{dh} para o valor da bola na posição (d, h). Uma linha "zig zag" pode passar do ponto (d, h) para o ponto $(d, h + 1)$ o $(d + 1, h - 1)$ caso tal ponto existe. Seja $V(d, h)$ o maior valor que podemos obter começando na diagonal d selecionando a bola na altura h. (Observe que neste caso todas bolas com altura menor que h não podem ser selecionadas, e todas bolas de altura h ou mais devem ser selecionadas.). Queremos encontrar o valor $V(0, 1)$ com diagonal "dummy" 0 de altura $N+1$ que consiste somente de bolas com valor 0. Os valores V satisfazem

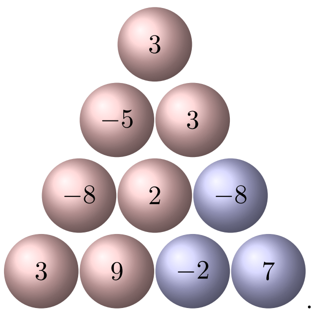
$$V(d,h) = \begin{cases} \max\{a_{dh} + V(d, h+1), \sum_{h \leq i \leq N-d+1} a_{di} + V(d+1, h-1)\} & h > 1 \text{ e } d < N \\ a_{dh} + V(d, h+1) & h = 1 \text{ e } d < N\text{, ou } d = N \end{cases}$$

Observe que a soma no primeiro caso da recorrência pode ser pré-calculada em tempo $O(N^2)$, logo temos um algoritmo com tempo total $O(N^2)$. A seguinta tabela mostra o nosso exemplo em coordenadas (d, h) na esquerda e os valor V na direita:

	5	0				
	4	0	3			
h	3	0	-5	3		
	2	0	-8	2	-8	
	1	0	3	9	-2	7
		0	1	2	3	4
			d			

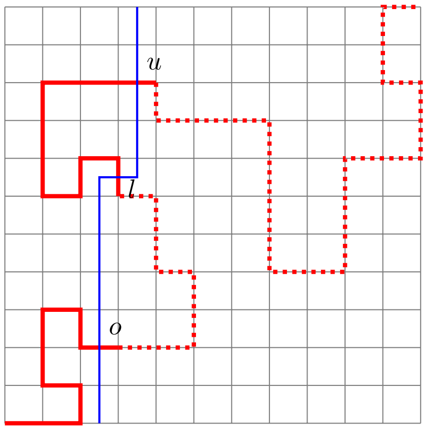
	5	6				
	4	6	6			
h	3	6	3	3		
	2	7	4	5	-1	
	1	7	7	14	-3	7
		0	1	2	3	4
			d			

Logo, no nosso exemplo o maior valor é 7 realizado pela seleção abaixo:

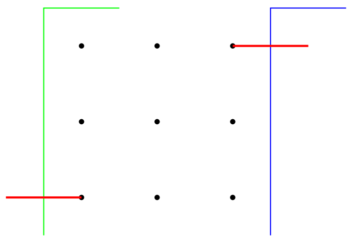


Exercício 5

A resposta é 1824132915142480492414708852236. Usando programação dinâmica isso pode ser calculado em tempo razoável[1]. Um subproblema é obtido cortando a grade entre duas colunas de vértices verticalmente, com uma aresta horizontal. Um dado caminho pode atravessar um tal corte várias vezes. Do ponto de vista do corte, isso gera um terminal aberto do caminho, e uma série de meio-laços, corretamente embutidas. Um exemplo de um corte na nossa instância, com um terminal aberto e um meio-laço:



Logo um corte é representado pela posição da aresta horizontal e rótulos que determinam como o caminho atravessa ele. Os rótulos são "o" (terminal aberto), "l" (terminal inferior de um meio-laço), "u" (terminal superior de um meio-laço) e "n" (não atravessa). O subproblema consiste em determinar o número de possibilidades de um caminho atravessar o corte para uma dada configuração (corte e rótulos). Este número pode ser obtido pelo número de caminhos atravessando o corte com a aresta horizontal uma posição acima, como mostrado na tabela na próxima página, em que "atual" representa uma configuração atual, e "próximo" as possíveis configurações sucessores. Partindo de uma configuração inicial, em que o caminho "entra" pelo lado inferior de baixo e calculando todas configurações sucessores até chegar numa configuração final em que o caminho "sai" pelo lado superior de cima, obtemos o número total de caminhos. Um exemplo para uma grade 3×3 com o corte inicial em verde e o corte final em azul e dois segmentes em vermelho que "entram" e "saiem"



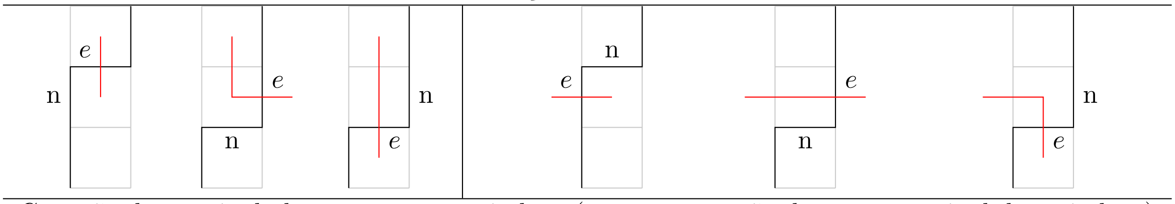
Temos que armazenar somente todas configurações de um dado corte e o rótulo "o" pode ocorrer no máximo uma vez, logo o algoritmo precisa memoria $O(3^n n)$ e tempo $O(3^n n^3)$ para processar todos n^2 vértices da grade. (Para um descrição mais detalhada ver, por exemplo, Bousquet-Mélou, Guttmann e Jensen [1].)

[1] Na minha máquina em 11s; o caso 12×12 precisa 45s, o caso 13×13 160s, o caso 14×14 564s. O fator limitante é o consumo de memoria, que provavelmente não permite mais que o caso 15×15 na minha máquina e com uma implementação simples que está disponível em http://www.inf.ufrgs.br/~mrpritt/saw.cpp.

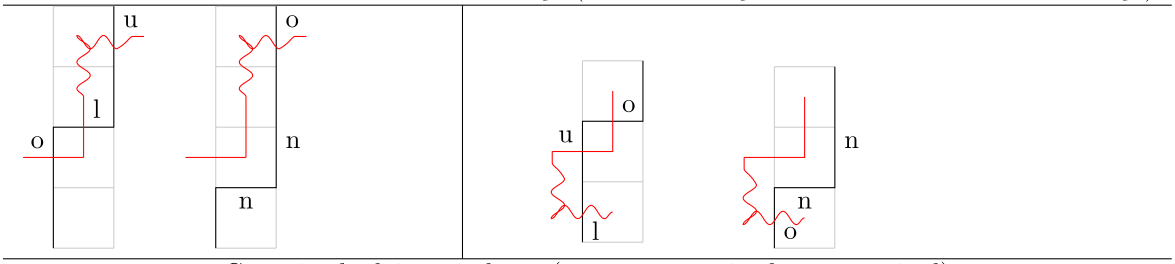
Atual Próximo Atual Próximo

Sem alteração / Criação de um meio-laço

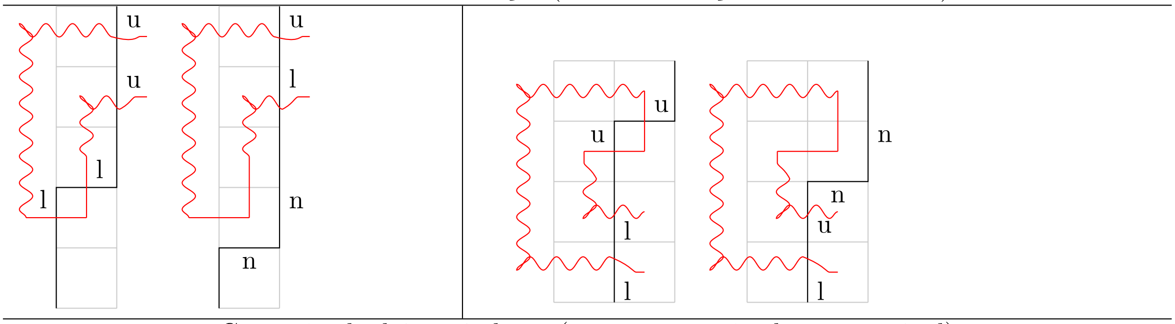
Continuação de um terminal



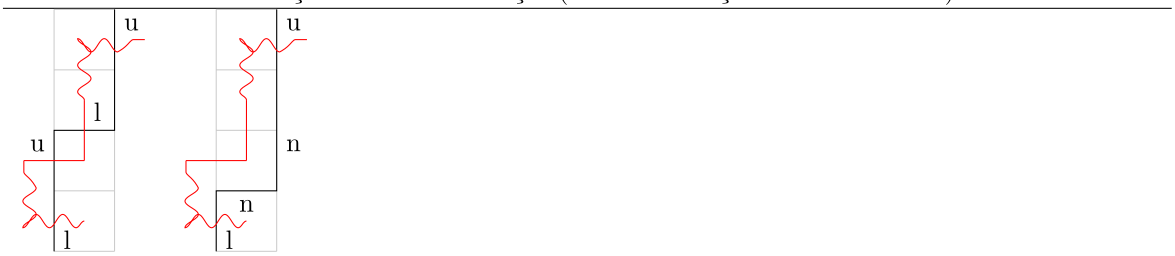
Conexão do terminal aberto com um meio-laço (com renomeação do outro terminal do meio-laço)



Conexão de dois meio-laços (com renomeação de um terminal)



Conexção de dois meio-laços (sem renomeação de um terminal)



(O rótulo e representa qualquer terminal $\{o, l, u\}$.)

Referências

[1] M. Bousquet-Mélou, A. J. Guttmann e I. Jensen. "Self-avoiding walks crossing a square". Em: *J. Physics A* 38.42 (2005). arXiv:cond-mat/0506341 [cond-mat.stat-mech].