OLPR0001

(12 de maio de 2016) – 5a.lista – Seleção de Variáveis e Domínios, PD, *Crosswords*

Fundamentos de Programação por Restrições

Joinville, 12 de maio de 2016

1 Será que Ele Avança?

Ranoberto e Ranubia são duas alegres rãs adolescentes que moram às margens do rio Cubatão, na Serra Dona Francisca. Ranoberto observou as longas pernas saltadoras de Ranubia e decidiu que quer conhecê-la melhor. Como ele é um pouco tímido e não sabe bem como iniciar a conversa, pensou em convidá-la para um jogo divertido, o que pode facilitar esta paquera. Ranubia gostou do estilo dele e aceitou o convite.

O jogo chama-se "Rã Saltadora": a partir de posições iniciais eles vão saltar um sobre o outro alternadamente. Ambos, Ranoberto e Ranubia, são capazes de saltar há uma distância horizontal máxima de até **10** unidades, em cada salto simples.

Você recebeu uma lista de posições válidas onde Ranoberto e Ranubia podem se posicionar: $x_1\ x_2\ ...x_n$. Como Ranoberto é um cavalheiro deixará para Ranubia o primeiro salto. Ranubia começa inicialmente na posição x_1 e Ranoberto começa inicialmente na posição x_2; o objetivo deles é alcançar a posição x_n. Determine o número mínimo de saltos necessários para que cada um, Ranoberto ou Ranubia, alcance o objetivo. Aos dois jogadores não é permitido permanecer na mesma posição ao mesmo tempo (afinal, eles ainda estão se conhecendo...), e em cada salto, o jogador que estiver atrás deve pular por cima do jogador à frente.

Entrada

O arquivo de entrada contém múltiplos casos de teste. Cada linha é um caso de teste e contém uma lista de inteiros $x_1\ x_2\ ...\ x_n$, onde $0 \leq x_1 < x_2... < x_n \leq 1000000$ (aqui era o problema original).

Saída

Para cada caso de teste de entrada, imprima o número total de saltos mínimo necessários para que um dos jogadores, ou Ranoberto ou Ranubia, chegue ao destino. Caso nenhum possa alcançar o destino, imprima `-1`.

Exemplo de Entrada	Exemplo de Saída
`3 5 9 12 15 17`	`3`
`3 5 9 12 30 40`	`-1`
`3 5`	`0`
`0 1 7 8 11 15 19`	`4`
`0 1 10 11 19 20`	`3`
`0 1 10 12 19 20`	`-1`

2 *Crossword Puzzle*

Consider the crossword puzzle shown in 1

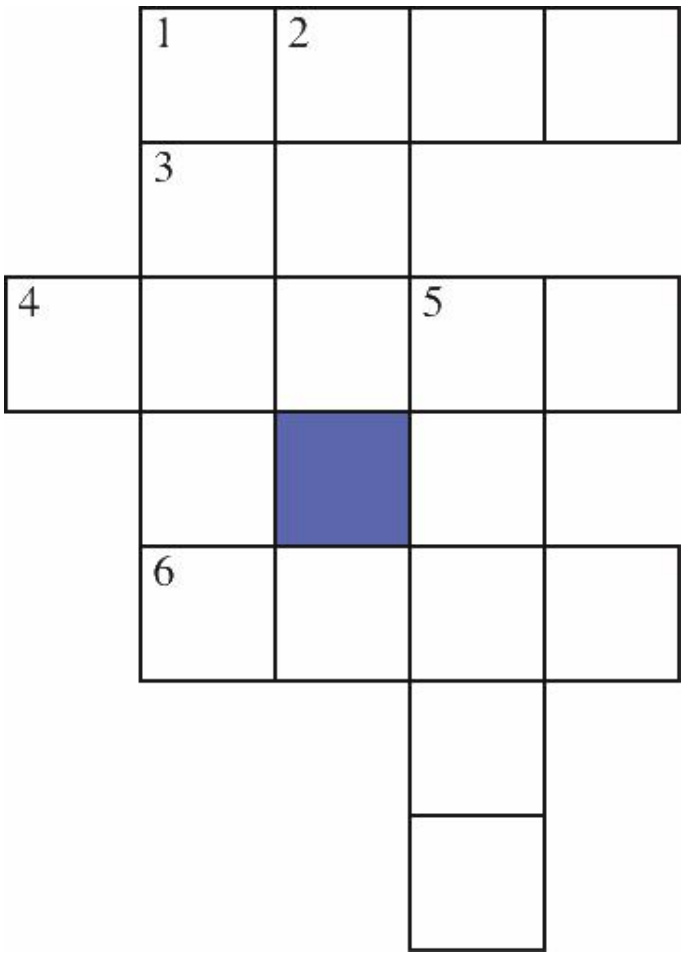

Figura 1: A crossword puzzle to be solved with seven words

The available words that can be used are:

at, eta, be, hat, he, her, it, him, on, one, desk, dance, usage, easy, dove, first, else, loses, fuels, help, haste, given, kind, sense, soon, sound, this, think.

1. Inicialmente dê uma solução para o problema acima, implemente e tenha as respostas.
2. Ok, feita a sua solução que atenda este problema, talvez seja ineficiente para instâncias maiores. Logo, pense utilizar alguma restrição global (tipo `table`, `element`, etc), ou criar uma nova restrição para estes problemas, afim de resolver de modo mais eficiente este problema.
3. Em seguida procure responder as questões que se seguem, pois há alternativas quanto as duas implementações que fizeste:

   (a) Given the representation with nodes for the positions (1-across, 2-down, etc.) and words for the domains, specify the network after domain consistency and arc consistency have halted.

(b) Consider the dual representation, in which the squares on the intersection of words are the variables and their domains are the letters that could go in these positions. Give the domains after this network has been made arc consistent. Does the result after arc consistency in this representation correspond to the result in part (a)?

(c) Show how variable elimination can be used to solve the crossword problem. Start from the arc-consistent network from part (a).

(d) Does a different elimination ordering affect the efficiency? Explain.

3 Problema das N-Rainhas em um Tabuleiro com Pesos (*weighted n-queens problem*)

O problema em questão é dado por: seja um tabuleiro de xadrez $N \times N$, onde cada uma das células tem um peso fixo. Determine uma configuração para posicionar N rainhas neste tabuleiro, tal que $\sum_{i=1}^{N} w_i$ seja o menor (ou o maior) valor possível selecionado. Veja figura 2.

Assim, sua tarefa é encontrar essa configuração de N de posicionar estas rainhas, sem estas se atacarem mutuamente, considerando os valores das células deste tabuleiro.

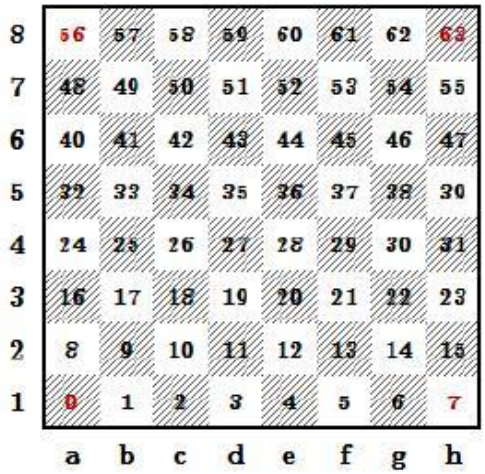

Figura 2: Tabuleiro de xadrez com pesos

3.1 Tarefa

Implemente este problema, contudo, rode para algumas instâncias ($N = 5, 10, 20, 40, 80$ – creio que até $N = 20$ teremos respostas em tempos aceitáveis) e procure preencher a tabela com os diversos tipos de escolha de variáveis e de domínio. Siga como exemplo a tabela abaixo (referência: tabelas 1 e 2) como guia, e escolha alguns testes do quais voce entende que para este problema vai alcançar um bons e péssimos resultados.

Tabela 1: Tempos de execução com manipulação dos parâmetros do *search*. $N = ...$

Sel. Variável / Atr. Dominio	`first_fail`	`anti_first_fail`	`input_order`
`indomain_min`	....	....	....
`indomain_max`	....	....	....
`indomain_median`	....	....	....
`indomain_split`	....	....	....
`indomain_random`	....	....	....
`indomain_interval`	....	....	....

3.2 Dicas de Implementação

O Minizinc oferece os parâmetros acima e detalhes do seu código podem estar em:

Tabela 2: Continuação da tabela anterior. $N = ...$

Sel. Variável / Atr. Dominio	largest	smallest	max_regret
indomain_min	....	....	....
indomain_max	....	....	....
indomain_median	....	....	....
indomain_split	....	....	....
indomain_random	....	....	....
indomain_interval	....	....	....

http://www.minizinc.org/g12_www/zinc/doc-2.0/html/zinc-manual.html#htoc17

Opções do *search* para busca com **inteiros** são resumidos em:

```
.......
%%solve :: int_search([LIST OF VARS], SELECT_VAR, DOMAIN_VAR, complete) minimize   CO
%% SELECT_VAR_INT={input_order, anti_first_fail, first_fail, smallest, largest, max_r
%% DOMAIN_VAR_INT={indomain, indomain_split, indomain_random, indomain_median, indoma
%% DETAILS in http://www.minizinc.org/g12_www/zinc/doc-2.0/html/zinc-manual.h
......
```

Preenchi nas tabelas 1 e 2

3.3 Alguns *Benchmarks*

1. Comece o *benchmark* abaixo:

```
      0   1   2   3   4   5   6   7   8   9  10

 0:   0   2   4   6   8  10   1   3   5   7   9
 1:   4   6   8  10   1   3   5   7   9   0   2
 2:   8  10   1   3   5   7   9   0   2   4   6
 3:   1   3   5   7   9   0   2   4   6   8  10
 4:   5   7   9   0   2   4   6   8  10   1   3
 5:   9   0   2   4   6   8  10   1   3   5   7
 6:   2   4   6   8  10   1   3   5   7   9   0
 7:   6   8  10   1   3   5   7   9   0   2   4
 8:  10   1   3   5   7   9   0   2   4   6   8
 9:   3   5   7   9   0   2   4   6   8  10   1
10:   7   9   0   2   4   6   8  10   1   3   5
```

2. Na sequência coloquei alguns *benchmarks* neste diretório, e em `http://www.cs.uky.edu/ai/pbmodels/`, há outros *benchmarks* difíceis.

3. Faça as entradas via arquivos, é direto e fácil.

Conclua o experimento, respondendo:

1. Porquê ocorreram os piores e melhores resultados para este problema em específico?

2. O que poderia ser feito para ser melhorado?

Se quiser estender o experimento acima e fazer uma *verdadeiro experimento científico*[1], fique confortável e será contabilizado na nota.

[1] Aumentar N, variar heurísticas, etc.

4 Considerações Finais:

➪ A modelagem do problema está naquelas 3 páginas comentadas na última aula. Há exemplos de uso do *search* no meu *github*
➪ Leia e siga as instruções de entrega
➪ Faça vários testes. Em geral ninguém faz, mas, é para fazer vários testes de I/Os
➪ Assuma e justifique os dados que faltarem.