SCC-218 Alg. Avançados e Aplicações

Lista 2 - Força Bruta e Backtracking (1)

Força Bruta

1. Considere um conjunto P de n pessoas e uma matriz M de tamanho $n \times n$, tal que $M[i,j] = M[j,i] = 1$, se as pessoas i e j se conhecem e $M[i,j] = M[j,i] = 0$, caso contrário.

   Problema: existe um subconjunto C (Clique), de r pessoas escolhidas de P, tal que qualquer par de pessoas de C se conhecem? Implemente uma solução força bruta. Veja a ilustração para melhor lhe orientar na implementação:

×	1	2	3	4	5	6	7	8
1	1	0	1	1	1	1	1	0
2	0	1	0	0	1	0	0	1
3	1	0	1	1	0	1	1	1
4	1	0	1	1	1	1	1	1
5	1	1	0	1	1	0	0	0
6	1	0	1	1	0	1	1	1
7	1	0	1	1	0	1	1	0
8	0	1	1	1	0	1	0	1

Figura 1: Conjunto P de 8 pessoas. Existe um conjunto C de 5 pessoas escolhidas de P tal que qualquer par de pessoas de C se conhecem?

2. Implemente um programa que enumere todas as combinações com repetições de tamanho r dentre um conjunto de n elementos.
3. Implemente um programa que enumere todos os arranjos simples (sem repetições) de tamanho r dentre um conjunto de n elementos.
4. Implemente um programa que enumere todos os arranjos com repetições de tamanho r dentre um conjunto de n elementos.
5. Dado um inteiro n, gere todas as possíveis senhas formadas por:
   - n dígitos
   - n dígitos ou letras minúsculas
   - n dígitos ou letras minúsculas ou letras maiúsculas

Backtracking

1. Dado um labrinto representado por uma matriz de tamanho $n \times m$, uma posição inicial $p_i = (x_i, y_i)$ e uma posição final $p_f = (x_f, y_f)$, tal que $p_i \neq p_f$, determinar se existe um caminho entre $p_i e p_f$.

   Podemos representar o labirinto como uma matriz M tal que:

$$M[x,y] = \begin{cases} -2, & \text{se a posição } (x,y) \text{ representa uma parede} \\ -1, & \text{se a posição}(x,y) \text{ nao pertence ao caminho} \\ i, & \text{tal que } i \geq 0, \text{ se a posição } (x,y) \text{ pertence ao caminho} \end{cases}$$

Neste caso, vamos supor que o labirinto é cercado por paredes, eventualmente apenas com exceção do local designado como saída.

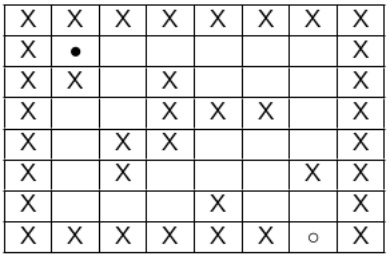

Figura 2: X é parede; ponto cheio é posição inicial e ponto vazio é pos final

Escreva um programa estilo backtracking que retorne o números de passos para se sair da origem e chegar ao destino. Obs: o problema pode nao ter solução.

2. Implemente o problema da Clique usando Backtracking.

3. Revisite o problema do Labirinto. Ao invés de determinar se existe um caminho entre o ponto inicial e o final (saída), encontre uma solução que usa o menor número possívell de passos.