Algoritmos e Programação de Computadores

Instituto de Computação

UNICAMP

Segundo Semestre de 2013

Força Bruta

- Força bruta (ou busca exaustiva) é um tipo de algoritmo de uso geral que consiste em enumerar todos os possíveis candidatos de uma solução e verificar se cada um satisfaz o problema.
- Esse tipo de algoritmo geralmente possui uma implementação simples e sempre encontrará uma solução se ela existir. Entretanto, seu custo computacional é proporcional ao número de candidatos a solução que, em problemas reais, tende a crescer exponencialmente.
- A força bruta é tipicamente usada em problemas cujo tamanho é limitado ou quando não se conhece um algoritmo mais eficiente.
- Também pode ser usado quando a simplicidade da implementação é mais importante do que a velocidade de execução, como nos casos de aplicações críticas em que os erros de algoritmo possuem sérias consequências.

Força Bruta - Clique

- 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?
- Solução de força bruta: verificar, para todas as combinações simples (sem repetições) C de r pessoas escolhidas entre as n pessoas do conjunto P, se todos os pares de pessoas de C se conhecem.

Força Bruta - Clique

- Considere um conjunto P de 8 pessoas representado pela matriz abaixo (de tamanho 8×8):

x	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

- Existe um conjunto C de 5 pessoas escolhidas de P tal que qualquer par de pessoas de C se conhecem?

Força Bruta - Clique × Combinação Simples

Existem 56 combinações simples de 5 elementos escolhidos dentre um conjunto de 8 elementos:

1 2 3 4 5	1 2 4 6 8	1 3 5 7 8	2 3 5 6 8
1 2 3 4 6	1 2 4 7 8	1 3 6 7 8	2 3 5 7 8
1 2 3 4 7	1 2 5 6 7	1 4 5 6 7	2 3 6 7 8
1 2 3 4 8	1 2 5 6 8	1 4 5 6 8	2 4 5 6 7
1 2 3 5 6	1 2 5 7 8	1 4 5 7 8	2 4 5 6 8
1 2 3 5 7	1 2 6 7 8	1 4 6 7 8	2 4 5 7 8
1 2 3 5 8	1 3 4 5 6	1 5 6 7 8	2 4 6 7 8
1 2 3 6 7	1 3 4 5 7	2 3 4 5 6	2 5 6 7 8
1 2 3 6 8	1 3 4 5 8	2 3 4 5 7	3 4 5 6 7
1 2 3 7 8	**1 3 4 6 7**	2 3 4 5 8	3 4 5 6 8
1 2 4 5 6	1 3 4 6 8	2 3 4 6 7	3 4 5 7 8
1 2 4 5 7	1 3 4 7 8	2 3 4 6 8	3 4 6 7 8
1 2 4 5 8	1 3 5 6 7	2 3 4 7 8	3 5 6 7 8
1 2 4 6 7	1 3 5 6 8	2 3 5 6 7	4 5 6 7 8

Força Bruta - Clique

Note que todos os pares de pessoas do subconjunto $C = \{1, 3, 4, 6, 7\}$ se conhecem:

x	1	3	4	6	7
1	1	1	1	1	1
3	1	1	1	1	1
4	1	1	1	1	1
6	1	1	1	1	1
7	1	1	1	1	1

Como enumerar todas as combinações simples de r elementos de um conjunto de tamanho n?

Força Bruta - Combinação Simples

```
#include <stdio.h>
#include <stdlib.h>

void combinacao_simples(int n, int r, int x[], int next, int k) {
  int i;

  if (k == r) {
    for (i = 0; i < r; i++)
      printf("%d ", x[i] + 1);
    printf("\n");
  } else {
    for (i = next; i < n; i++) {
      x[k] = i;
      combinacao_simples(n, r, x, i + 1, k + 1);
    }
  }
}
```

Força Bruta - Combinação Simples

```
int main() {
  int n, r, *x;

  printf("Entre com o valor de n: ");
  scanf("%d", &n);

  printf("Entre com o valor de r: ");
  scanf("%d", &r);

  x = malloc(r * sizeof(int));

  combinacao_simples(n, r, x, 0, 0);

  free(x);

  return 0;
}
```

Força Bruta - Ciclo Hamiltoniano

- Considere um conjunto de n cidades e uma matriz M de tamanho $n \times n$ tal que $M[i, j] = 1$, se existe um caminho direto entre as cidades i e j, e $M[i, j] = 0$, caso contrário.
- Problema: existe uma forma de, saindo de uma cidade qualquer, visitar todas as demais cidades, sem passar duas vezes por nenhuma cidade, e ao final retornar para a cidade inicial?
- Note que, se existe uma forma de sair de uma cidade X qualquer, visitar todas as demais cidades (sem repetir nenhuma) e depois retornar para X, então existe uma forma de fazer o mesmo para qualquer outra cidade do conjunto, já que existe um Ciclo Hamiltoniano (uma forma circular de visitar todas as cidades) e qualquer cidade do ciclo pode ser usado como ponto de partida.

Força Bruta - Ciclo Hamiltoniano

- Como vimos, qualquer cidade pode ser escolhida como cidade inicial. Sendo assim, vamos escolher, arbitrariamente a cidade n como ponto de partida.
- Solução de força bruta: testar todas as permutações das $n-1$ primeiras cidades, verificando se existe um caminho direto entre a cidade n e a primeira da permutação, assim como um caminho entre todas as cidades consecutivas da permutação e, por fim, um caminho direto entre a última cidade da permutação e a cidade n.
- Ciclo Hamiltoniano: $n \rightsquigarrow [p_1 \rightsquigarrow p_2 \rightsquigarrow p_3 \rightsquigarrow \cdots \rightsquigarrow p_{n-1}] \rightsquigarrow n$.

Força Bruta - Ciclo Hamiltoniano

- Considere um conjunto de 8 cidades representado pela matriz abaixo (de tamanho 8×8):

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

- Existe uma forma de, a partir da cidade 8, visitar todas as demais cidades, sem repetir nenhuma, e ao final retornar para a cidade 8?

Força Bruta - Ciclo Hamiltoniano × Permutação

Exitem 5040 permutações das 7 primeiras cidades da lista original:

1 2 3 4 5 6 7	···	7 6 5 2 3 4 1
1 2 3 4 5 7 6	3 6 4 5 1 7 2	7 6 5 2 4 1 3
1 2 3 4 6 5 7	3 6 4 5 2 1 7	7 6 5 2 4 3 1
1 2 3 4 6 7 5	3 6 4 5 2 7 1	7 6 5 3 1 2 4
1 2 3 4 7 5 6	3 6 4 5 7 1 2	7 6 5 3 1 4 2
1 2 3 4 7 6 5	3 6 4 5 7 2 1	7 6 5 3 2 1 4
1 2 3 5 4 6 7	3 6 4 7 1 2 5	7 6 5 3 2 4 1
1 2 3 5 4 7 6	**3 6 4 7 1 5 2**	7 6 5 3 4 1 2
1 2 3 5 6 4 7	3 6 4 7 2 1 5	7 6 5 3 4 2 1
1 2 3 5 6 7 4	3 6 4 7 2 5 1	7 6 5 4 1 2 3
1 2 3 5 7 4 6	3 6 4 7 5 1 2	7 6 5 4 1 3 2
1 2 3 5 7 6 4	3 6 4 7 5 2 1	7 6 5 4 2 1 3
1 2 3 6 4 5 7	3 6 5 1 2 4 7	7 6 5 4 2 3 1
1 2 3 6 4 7 5	3 6 5 1 2 7 4	7 6 5 4 3 1 2
1 2 3 6 5 4 7	···	7 6 5 4 3 2 1

Como enumerar todas as permutações de n valores distintos?

Força Bruta - Permutação

```
#include <stdio.h>
#include <stdlib.h>

void permutacao(int n, int x[], int used[], int k) {
  int i;
  if (k == n) {
    for (i = 0; i < n; i++)
      printf("%d ", x[i] + 1);
    printf("\n");
  } else {
    for (i = 0; i < n; i++)
      if (!used[i]) {
        used[i] = 1;
        x[k] = i;
        permutacao(n, x, used, k + 1);
        used[i] = 0;
      }
  }
}
```

Força Bruta - Permutação

```
int main() {
  int i, n, *x, *used;

  printf("Entre com o valor de n: ");
  scanf("%d", &n);

  x = malloc(n * sizeof(int));
  used = malloc(n * sizeof(int));

  for (i = 0; i < n; i++)
    used[i] = 0;

  permutacao(n, x, used, 0);

  free(x);
  free(used);

  return 0;
}
```

Força Bruta - Exercícios

Exercício

Implemente um programa que resolva o problema da Clique, usando força bruta.

Exercício

Implemente um programa que resolva o problema do Ciclo Hamiltoniano, usando força bruta.

Força Bruta - Exercícios

Exercício

Implemente um programa que enumere todas as combinações com repetições de tamanho r dentre um conjunto de n elementos.

Exercício

Implemente um programa que enumere todos os arranjos simples (sem repetições) de tamanho r dentre um conjunto de n elementos.

Exercício

Implemente um programa que enumere todos os arranjos com repetições de tamanho r dentre um conjunto de n elementos.

Força Bruta - Exercícios

Exercício

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*

Força Bruta - Combinação com Repetições

```
#include <stdio.h>
#include <stdlib.h>

void combinacao_repeticao(int n, int r, int x[],  int used, int k) {
  int i;

  if (k == r) {
    for (i = 0; i < r; i++)
      printf("%d ", x[i] + 1);
    printf("\n");
  } else {
    for (i = used; i < n; i++) {
      x[k] = i;
      combinacao_repeticao(n, r, x, i, k + 1);
    }
  }
}
```

Força Bruta - Combinação com Repetições

```
int main() {
  int n, r, *x;

  printf("Entre com o valor de n: ");
  scanf("%d", &n);

  printf("Entre com o valor de r: ");
  scanf("%d", &r);

  x = malloc(r * sizeof(int));

  combinacao_repeticao(n, r, x, 0, 0);

  free(x);

  return 0;
}
```

Força Bruta - Arranjo Simples

```
#include <stdio.h>
#include <stdlib.h>

void arranjo_simples(int n, int r, int x[], int used[], int k) {
  int i;

  if (k == r) {
    for (i = 0; i < r; i++)
      printf("%d ", x[i] + 1);
    printf("\n");
  } else {
    for (i = 0; i < n; i++) {
      if (!used[i]) {
        used[i] = 1;
        x[k] = i;
        arranjo_simples(n, r, x, used, k + 1);
        used[i] = 0;
      }
    }
  }
}
```

Força Bruta - Arranjo Simples

```
int main() {
  int i, n, r, *x, *used;

  printf("Entre com o valor de n: ");
  scanf("%d", &n);

  printf("Entre com o valor de r: ");
  scanf("%d", &r);

  x = malloc(r * sizeof(int));
  used = malloc(n * sizeof(int));

  for (i = 0; i < n; i++)
    used[i] = 0;

  arranjo_simples(n, r, x, used, 0);

  free(x);
  free(used);

  return 0;
}
```

Força Bruta - Arranjo com Repetições

```
#include <stdio.h>
#include <stdlib.h>

void arranjo_repeticao(int n, int r, int x[], int k) {
  int i;

  if (k == r) {
    for (i = 0; i < r; i++)
      printf("%d ", x[i] + 1);
    printf("\n");
  } else {
    for (i = 0; i < n; i++) {
      x[k] = i;
      arranjo_repeticao(n, r, x, k + 1);
    }
  }
}
```

Força Bruta - Arranjo com Repetições

```
int main() {
  int n, r, *x;

  printf("Entre com o valor de n: ");
  scanf("%d", &n);

  printf("Entre com o valor de r: ");
  scanf("%d", &r);

  x = malloc(r * sizeof(int));

  arranjo_repeticao(n, r, x, 0);

  free(x);

  return 0;
}
```