ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

LẬP TRÌNH C CƠ BẢN

Chương 2. Đệ quy

Soạn bởi: TS. Nguyễn Bá Ngọc

(Dựa trên bài giảng sử dụng chung)

2021

NỘI DUNG

- Đệ quy & chia để trị
- Đệ quy có nhớ
- Đệ quy quay lui

NỘI DUNG

- Đệ quy & chia để trị
- Đệ quy có nhớ
- Đệ quy quay lui

Đệ quy

- Trong C: Câu lệnh gọi hàm được thực hiện trong thời gian xử lý của chính hàm đó được gọi là gọi hàm đệ quy. NNLT C hỗ trợ đồng thời cả gọi hàm đệ quy trực tiếp và gián tiếp.
- Đáy đệ quy: Là trường hợp xử lý hàm đệ quy kết thúc mà không dẫn đến lệnh gọi hàm đệ quy.

rsum là 1 hàm đệ quy

Đáy đệ quy

Lệnh gọi hàm đệ quy

vd2-1.c

```c
#include <stdio.h>
int rsum(int n) {
  if (n == 1) {
    return 1;
  }
  return n + rsum(n - 1);
}
int main() {
  printf("n = ");
  int n;
  scanf("%d", &n);
  printf("%d\n", rsum(n));
  return 0;
}
```

Chia để trị

- Phương pháp giải quyết bài toán bằng cách chia nhỏ bài toán thành những bài toán con nhỏ hơn. Các bài toán con lại tiếp tục được chia nhỏ thành những bài toán con nhỏ hơn nữa. Tiến trình chia nhỏ được thực hiện cho tới khi các bài toán con thu được đủ nhỏ và đã có lời giải cho những bài toán con đó. Lời giải của các bài toán con sau đó được tổng hợp lại thành lời giải của bài toán ban đầu theo trình tự ngược với trình tự chia nhỏ.
  - Những bài toán con đủ nhỏ và đã có lời giải (vì vậy không cần chia nhỏ hơn nữa) được gọi là các trường hợp cơ sở của bài toán.
  - Cấu trúc chia nhỏ bài toán thường được biểu diễn bằng công thức đệ quy.

Ví dụ 2.1. Chia để trị

- Xét bài toán tính tổng các số nguyên dương từ 1 tới n.
- Chúng ta ký hiệu s(n) = 1 + 2 + ... + n, trong đó n là số nguyên dương:
  - Chúng ta có thể tính s(n) với n > 1 thông qua s(n - 1) bằng công thức s(n) = n + s(n - 1).
  - Với n = 1 thì s(n) = 1.
- Ví dụ để tính s(5) chúng ta tính 5 + s(4)

  Để tính s(4) chúng ta tính 4 + s(3)

  Để tính s(3) chúng ta tính 3 + s(2)

  Để tính s(2) chúng ta tính 2 + s(1)

  Chúng ta biết s(1) = 1 =>

  5 + 9 = 14

  4 + 5 = 9

  3 + 2 = 5

  2 + 1 = 3

  - Kết quả cuối cùng: s(5) = 14

Ví dụ chỉ mang tính minh họa cho phương pháp chia để trị.
*Tất nhiên chúng ta có thể tính s(n) = n * (n + 1)/2*

Đệ quy & Chia để trị

- Một giải thuật được thiết kế bằng phương pháp chia để trị có thể được triển khai bằng đệ quy.
  - Tất nhiên có thể triển khai theo cách khác không sử dụng đệ quy và hiệu quả hơn đệ quy (ví dụ bằng vòng lặp).
  - Tuy nhiên đệ quy là một cách khá đơn giản để triển khai giải thuật chia để trị.
    - Công thức đệ quy <=> Gọi hàm đệ quy
    - Trường hợp cơ sở <=> Đáy đệ quy

Ví dụ bài toán tính tổng dãy 1 + 2 + … + n

$$s(n) = \begin{cases} 1, \text{ nếu } n == 1 \\ n + s(n - 1), \text{ nếu ngược lại } (n > 1) \end{cases}$$

Cấu trúc chia nhỏ bài toán & triển khai đệ quy

vd2-1.c

```c
#include <stdio.h>
int rsum(int n) {
  if (n == 1) {
    return 1;
  }
  return n + rsum(n - 1);
}
int main() {
  printf("n = ");
  int n;
  scanf("%d", &n);
  printf("%d\n", rsum(n));
  return 0;
}
```

Ví dụ 2.2. Tính số Fibonacci

- Dãy Fibonacci:
  - 1, 1, 2, 3, 5, 8, 13, 21, 34, …
  - Sử dụng chỉ số từ 0

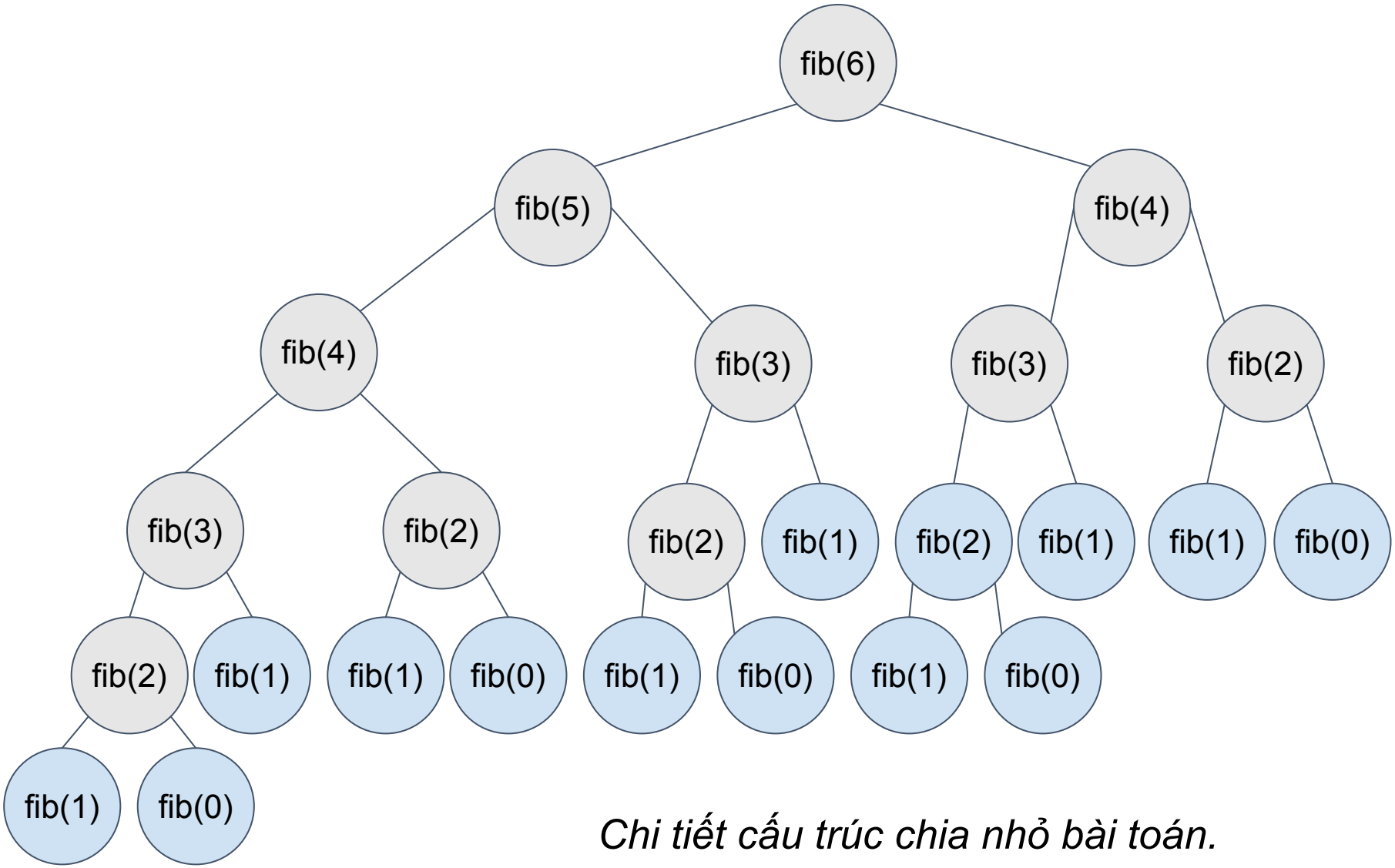
Cấu trúc chia nhỏ bài toán

$$Fib(n) = \begin{cases} 1, \text{ nếu } n == 0 \,||\, n == 1 \\ Fib(n-1) + Fib(n-2), \\ \text{nếu ngược lại } (n > 1) \end{cases}$$

vd2-2.c

```c
#include <stdio.h>
int fib(int n) {
  if (n == 0 || n == 1) {
    return 1;
  }
  return fib(n - 1) + fib(n - 2);
}
int main() {
  printf("Nhập n = ");
  int n;
  scanf("%d", &n);
  printf("Fib(%d) = %d\n", n, fib(n));
  return 0;
}
```

Ví dụ 2.2. Tính số Fibonacci(2)



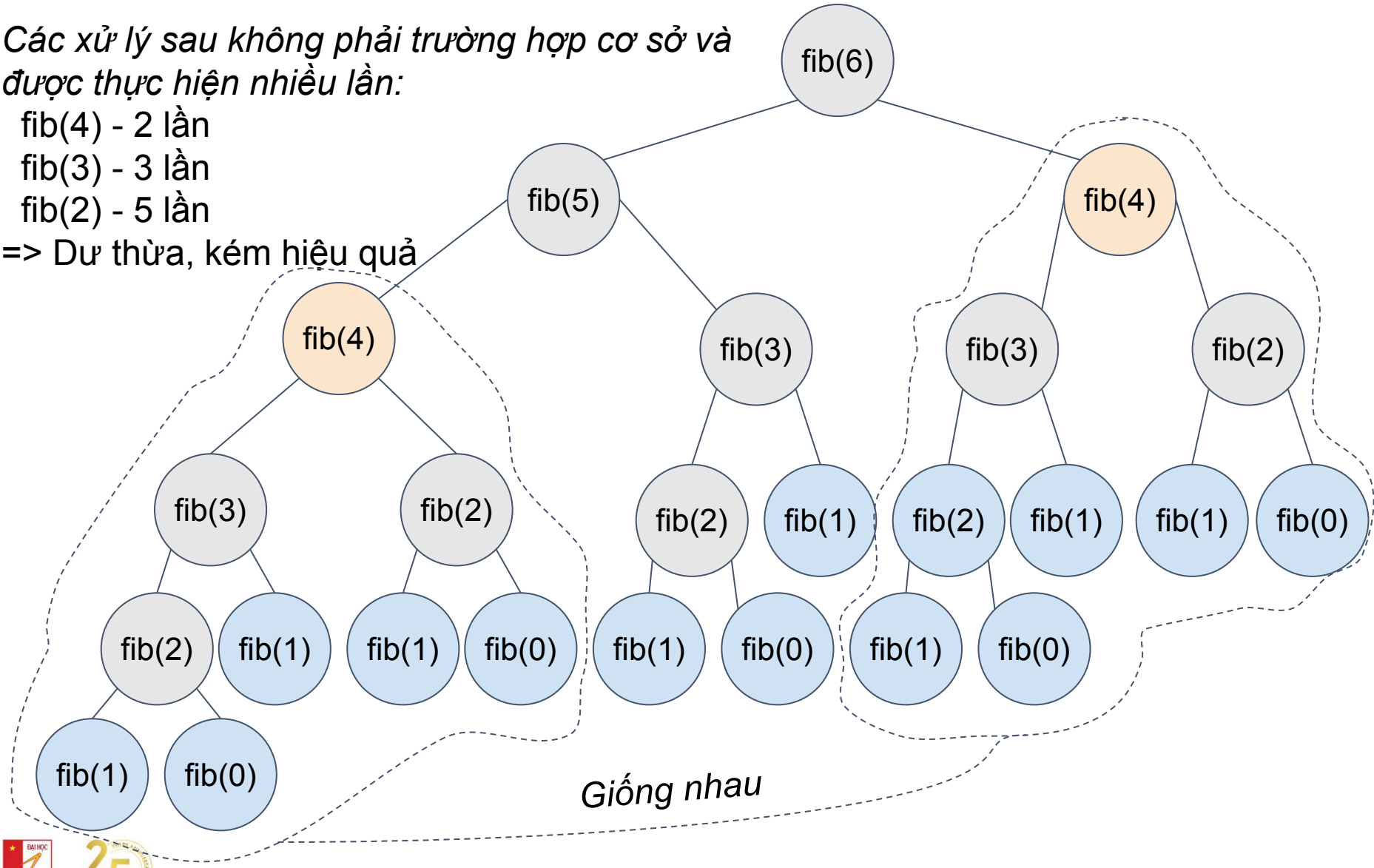
Chi tiết cấu trúc chia nhỏ bài toán.

Vấn đề chồng lấn bài toán con

Các xử lý sau không phải trường hợp cơ sở và được thực hiện nhiều lần:

fib(4) - 2 lần
fib(3) - 3 lần
fib(2) - 5 lần

=> Dư thừa, kém hiệu quả



Vấn đề chồng lấn bài toán con (2)

- Chồng lấn bài toán con là trường hợp trong tiến trình chia nhỏ bài toán ban đầu có những bài toán con được giải nhiều lần.
  - Bài toán con được giải nhiều lần với các bước giống hệt nhau => dư thừa và tốn thời gian tính toán vô ích => tiềm năng tối ưu hóa.
- Phương hướng loại bỏ dư thừa và tránh lặp lại các tính toán không cần thiết:
  - Quy hoạch động (không học trong môn này nên không phân tích ở đây)
  - => Sử dụng bộ nhớ để lưu lời giải cho các bài toán con
    - Nếu bài toán con đã được giải thì trả về kết quả đã ghi nhớ mà không cần thực hiện các tính toán tiếp theo. Trong trường hợp ngược lại thì thực hiện các tính toán và ghi nhớ lời giải để sử dụng cho những lần tiếp theo.
    - => Đệ quy có nhớ.

NỘI DUNG

- Đệ quy & chia để trị
- Đệ quy có nhớ
- Đệ quy quay lui

Ví dụ 2-3. Đệ quy có nhớ

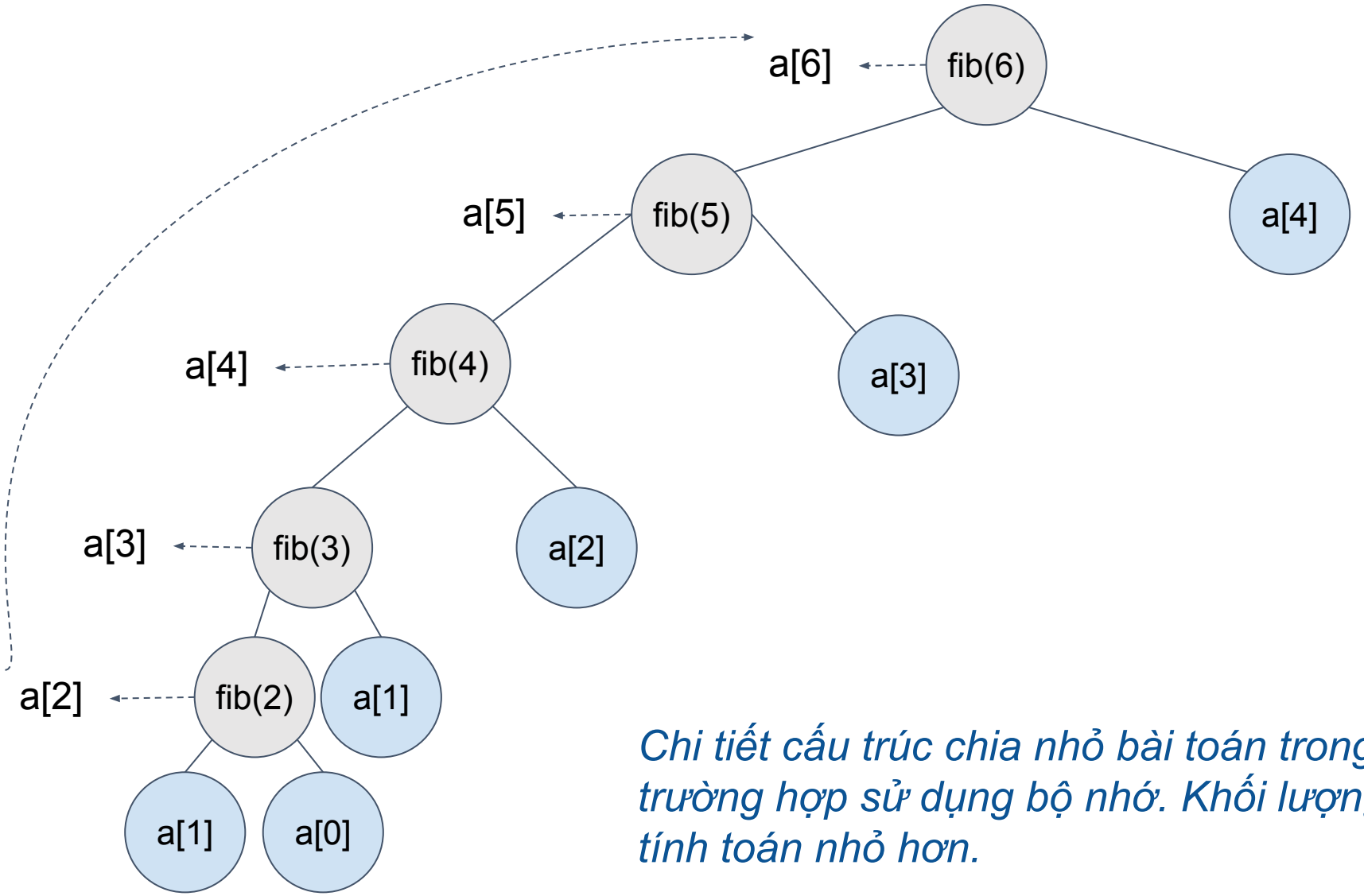
Bộ nhớ

vd2-3.c

```c
#include <stdio.h>
int a[100] = {1, 1};
int fib(int n) {
  if (a[n]) {
    return a[n];
  }
  return (a[n] = fib(n - 1) + fib(n - 2));
}
int main() {
  printf("Nhập n = ");
  int n;
  scanf("%d", &n);
  printf("Fib(%d) = %d\n", n, fib(n));
  return 0;
}
```

Mục đích của ví dụ này là để minh họa đệ quy có nhớ. Tất nhiên có thể tính số Fibonacci theo cách hiệu quả hơn bằng vòng lặp.

Ví dụ 2-3. Đệ quy có nhớ (2)



Chi tiết cấu trúc chia nhỏ bài toán trong trường hợp sử dụng bộ nhớ. Khối lượng tính toán nhỏ hơn.

Ví dụ 2-4. Tính số lượng tổ hợp

Cấu trúc chia nhỏ bài toán

$$C(k,n) = \begin{cases} 1, \text{ nếu } k == 0 \,||\, k == n; \\ C(k-1, n-1) + C(k, n-1), \\ \text{nếu ngược lại } (0< k < n). \end{cases}$$

k	0	1	2	3	4	5	6	n
	1	1						1
	1	2	1					2
	1	3	3	1				3
	1	4	6	4	1			4
	1	5	10	10	5	1		5
	1	6	15	20	15	6	1	6

vd2-4.c

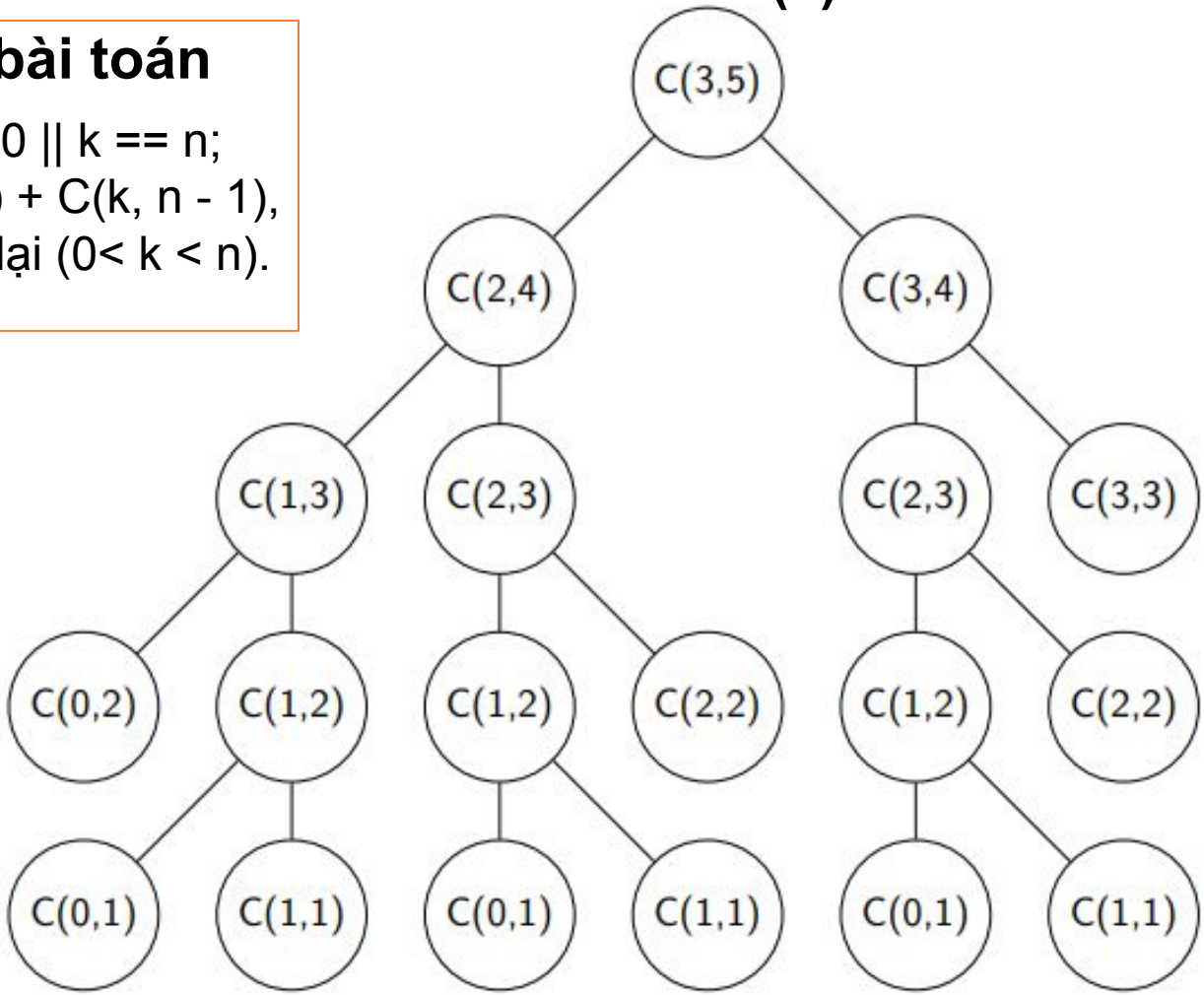
```c
#include <stdio.h>
int combinations(int k, int n) {
  if (k == 0 || k == n) {
    return 1;
  }
  return combinations(k - 1, n - 1) +
         combinations(k, n - 1);
}
int main() {
  printf("Nhập k, n: ");
  int k, n;
  scanf("%d%d", &k, &n);
  printf("%d\n", combinations(k, n));
  return 0;
}
```

Tất nhiên chúng ta có thể tính C(k, n) = n!/(n-k)!/k! *khi cần.*
Ví dụ chỉ mang tính chất minh họa cho phương pháp chia để trị và đệ quy.

Ví dụ 2-4. Tính số lượng tổ hợp (2)

Cấu trúc chia nhỏ bài toán

$$C(k,n) = \begin{cases} 1, \text{ nếu } k == 0 \,||\, k == n; \\ C(k-1, n-1) + C(k, n-1), \\ \text{nếu ngược lại } (0< k < n). \end{cases}$$



Có chồng lấn bài toán con hay không?

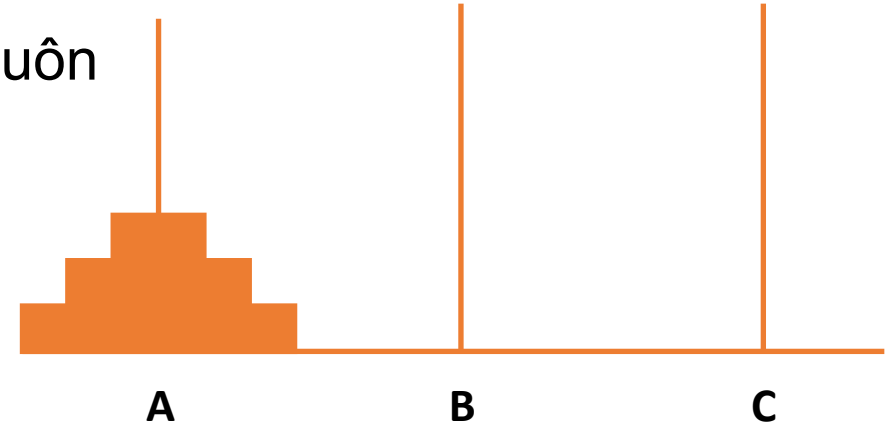
Bài tập 2.1. Đệ quy có nhớ

Điều chỉnh ví dụ 2.4 tính số lượng tổ hợp bằng đệ quy, sử dụng bộ nhớ để lưu lời giải cho các bài toán con.

Gợi ý: c[N][N]

Bài toán tháp Hà Nội

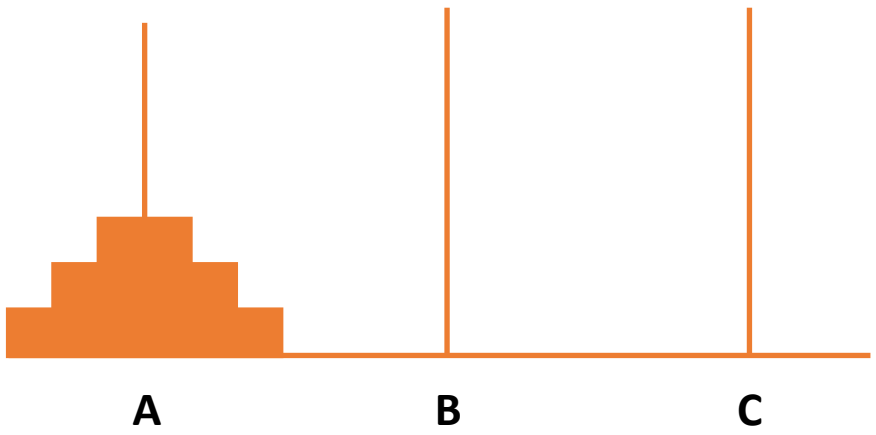
- Có n đĩa với kích thước khác nhau và 3 cọc A, B, C
  - *Đĩa có lỗ ở chính tâm.*
- Ban đầu n đĩa được xếp thành chồng ở cọc A, đĩa lớn hơn nằm dưới và đĩa nhỏ hơn nằm trên.
- Yêu cầu bài toán: Đưa ra chỉ dẫn chuyển từng đĩa một cách tuần tự để di chuyển chồng đĩa từ cọc A sang cọc B.
  - Mỗi lần chỉ được di chuyển 1 đĩa.
  - Đảm bảo đĩa nhỏ hơn luôn luôn nằm trên đĩa lớn hơn.



Bài toán tháp Hà Nội(2)

- Ví dụ lời giải cho n = 3:

  B1: A □ B
  B2: A □ C
  B3: B □ C
  B4: A □ B
  B5: C □ A
  B6: C □ B
  B7: A □ B



Cấu trúc chia nhỏ bài toán. Để chuyển n đĩa từ A sang B: Nếu n > 1 thì chuyển n - 1 đĩa từ A sang C, sau đó chuyển 1 đĩa từ A sang B, và cuối cùng chuyển n - 1 đĩa từ C sang A; nếu ngược lại (n==1) thì xuất A => B.

Gợi ý: move(n, a, b, c)

move(n - 1, a, c, b)

Xuất: a =>b

move(n - 1, c, b, a)

Bài tập 2.2. Bài toán tháp Hà Nội

Cài đặt chương trình giải bài toán tháp Hà Nội:

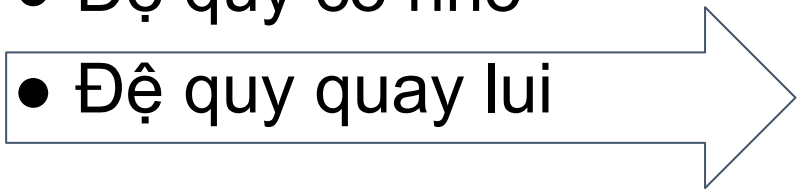
- Chương trình hỏi người dùng nhập vào số n là số lượng đĩa ban đầu trên cọc A.
- Chương trình tính và xuất ra màn hình số bước di chuyển cần thực hiện để chuyển chồng đĩa sang cọc B.
- Nếu số bước di chuyển > 1000 thì chương trình hỏi người dùng có muốn xem chi tiết các bước hay không? Nếu người dùng chọn có thì chương trình xuất ra chỉ dẫn chuyển đĩa. Nếu ngược lại thì kết thúc chương trình.

Gợi ý: Cài đặt bằng đệ quy.

Tất nhiên có thể tìm hiểu thêm về cách giải bằng vòng lặp.

NỘI DUNG

- Đệ quy & chia để trị
- Đệ quy có nhớ
- Đệ quy quay lui



Đệ quy quay lui

- Phương pháp giải quyết bài toán bằng cách xây dựng dần lời giải qua một chuỗi tuần tự các bước đệ quy, mỗi bước đệ quy lần lượt thử các lựa chọn cho một thành phần trong lời giải bài toán. Có thể bỏ qua các lựa chọn nếu có thể chứng minh các lựa chọn đó không dẫn đến lời giải cần tìm ở thời điểm bất kỳ. Sau khi duyệt hết các lựa chọn ở một bước thì quay lại bước trước đó.
  - Bỏ qua các lựa chọn không có khả năng dẫn đến lời giải cần tìm giúp cắt giảm thời gian tính toán. Kỹ thuật này còn được gọi là cắt tỉa không gian tìm kiếm hoặc nhánh cận.

Bài toán tiêu biểu cho đệ quy quay lui

Các bài toán tiêu biểu cho đệ quy quay lui có thể được mô hình hóa như sau:

- Lời giải bài toán bao gồm n thành phần được ký hiệu là $x_0, x_1,\ldots,x_{n-1}$.
- Chúng ta ký hiệu:
  - A_i, i trong khoảng [0, n), là tập hợp các lựa chọn cho thành phần x_i;
  - $A = \{(x_0, x_1, ..., x_{n-1})|x_i \in A_i, \forall\ i = 0, ..., n-1\}$ là không gian lời giải bài toán (tập hợp tất cả các bộ giá trị/lời giải tiềm năng).
- Các yêu cầu tiêu biểu:
  - Liệt kê tất cả các bộ giá trị;
  - Tìm bộ giá trị tối ưu;
  - Liệt kê các bộ giá trị thỏa mãn điều kiện (sàng lọc).

Giải thuật đệ quy quay lui

- Giải thuật đệ quy quay lui có cấu trúc khái quát như sau
- Hàm solve(*i*):
  - Với mỗi giá trị *v* trong miền giá trị A_i của x_i chúng ta kiểm tra khả năng đáp ứng yêu cầu bài toán nếu gán x_i = v.
  - Nếu *v* không thể đáp ứng yêu cầu thì bỏ qua;
  - Nếu ngược lại thì:
    - Gán x_i = v;
    - Nếu *i* < *n - 1* thì gọi đệ quy solve(*i*+1);
    - Nếu ngược lại (*i* == *n - 1)* thì thực hiện xử lý đầu ra với lời giải tiềm năng hiện có, kiểm tra xem có phải là lời giải đang cần tìm hay không.

(nhánh cận)

- Bắt đầu với solve(0).

Giải thuật đệ quy quay lui (2)

Mã giả C cho giải thuật đệ quy quay lui khái quát, bắt đầu với solve(0):

```
void solve(i) {
    for each v in A[i] {
       if is_candidate(v,i) {
          x[i] = v;
          if(i == n - 1) {
              process_output();
           } else {
              solve(i + 1);
           }
       }
}
```

Ví dụ 2-5. Liệt kê xâu nhị phân

- Bài toán: Liệt kê tất cả các dãy số nhị phân có n bits.
- Biểu diễn lời giải:
  - Mảng x gồm n phần tử với các chỉ số i trong khoảng [0, n)
  - Mỗi phần tử x[i] tương ứng với bit thứ i trong dãy.
  - Mỗi phần tử x[i] chỉ có thể nhận giá trị là 0 hoặc 1.

vd2-5.c

```c
#include <stdio.h>
#define N 100
char x[N];
int n;
void output() {
  for (int i = 0; i < n; ++i) {
    printf("%d", (int)x[i]);
  }
  printf("\n");
}
void solve(int i) {
  for (char v = 0; v < 2; ++v) {
    x[i] = v;
    if (i == n - 1) {
      output();
    } else {
      solve(i + 1);
    }
  }
}
int main() {
  printf("Nhap n = ");
  scanf("%d", &n);
  solve(0);
  return 0;
}
```

Bài tập 2.3. Dãy số nhị phân 11

- Viết chương trình liệt kê tất cả các dãy số nhị phân gồm n bits sao cho không có 2 bits liên tiếp nào có giá trị giống nhau.

- Gợi ý:
  - i < 1 || x[i] != x[i - 1]
  - Có thể áp dụng nhánh cận để giải thuật hoạt động nhanh hơn.

Bài tập 2.4. Dãy số nhị phân 101

- Viết chương trình liệt kê tất cả các dãy số nhị phân gồm n bits sao cho dãy không chứa cụm bits 101.

- Gợi ý:
  - i < 2 || (x[i - 2] != 1 && x[i - 1] != 0 && x[i] != 1)
  - Có thể áp dụng nhánh cận để giải thuật hoạt động nhanh hơn.

Bài tập 2.5. Dãy số nhị phân k bits 1

- Viết chương trình liệt kê tất cả các dãy số nhị phân gồm n bits sao cho dãy chứa đúng k bits 1 với k là số nguyên và k <= n.

- Gợi ý:
  - sum = k
  - Có thể áp dụng nhánh cận để hoạt động nhanh hơn.

Bài toán liệt kê tổ hợp

- Liệt kê các tổ hợp chập *k* của n số 0, 1, …, n-1
  - Trong đó 0 < k <=n.
- Cấu trúc phát triển lời giải:
  - Chúng ta biểu diễn tổ hợp chập k của n phần tử bằng mảng x gồm k phần tử và có thể quy ước x[0] < x[1] < … < x[k - 1] mà không làm thay đổi ý nghĩa bài toán (không bỏ qua bất kỳ lời giải nào).
  - Lần lượt chọn giá trị cho từng phần tử của mảng x.
    - Bắt đầu với i = 0.
    - Có n - k + 1 cách chọn giá trị cho x[0], A_0 = {0, 1, …, n - k}, giá trị cực đại của x[0] là n - k bởi vì chúng ta cần k - 1 giá trị cho phần còn lại.
    - Với i > 0, giá trị của x[i] phải thỏa mãn điều kiện x[i] > x[i - 1], vì vậy A_i = {x[i - 1] + 1, …, n - k + i]
    - Nếu i == k - 1 thì xuất giá trị tổ hợp, nếu ngược lại thì tiếp tục gọi đệ quy với i + 1.

Vi dụ 2.6. Liệt kê tổ hợp chập k của n

vd2-6.c

```c
#include <stdio.h>
#define N 100
int x[N], k, n;
void output() {
  for (int i = 0; i < k; ++i) {
    printf("%d", x[i]);
  }
  printf("\n");
}
void solve(int i) {
  int v = i > 0? x[i - 1] + 1: 0;
  for (; v <= n - k + i; ++v) {
    x[i] = v;
    if (i == k - 1) {
      output();
    } else {
      solve(i + 1);
    }
  }
}
int main() {
  printf("Nhập k, n: ");
  scanf("%d%d", &k, &n);
  solve(0);
  return 0;
}
```

Duyệt các giá trị cho thành phần thứ i của lời giải

Xử lý đầu ra

Mở rộng lời giải

Bài tập 2.6. Tổng lớn nhất

- Viết chương trình nhập vào một dãy n số nguyên.
- Nhập vào một số k < n.
- Tìm k số trong dãy n số nguyên được nhập vào có tổng lớn nhất. Có thể đưa ra bộ k số bất kỳ nếu có nhiều đáp án.

- Gợi ý:
  - Có thể phát triển ví dụ 2-6, kiểm tra từng bộ giá trị (*vét cạn*).
  - *Tất nhiên có thể làm theo cách khác.*

Bài tập 2.7. Tích lớn nhất

- Viết chương trình nhập vào một dãy n số nguyên.
- Nhập vào một số k < n.
- Tìm k số trong dãy n số nguyên được nhập vào có tích lớn nhất. Có thể đưa ra bộ k số bất kỳ.

- Gợi ý:
  - Có thể phát triển ví dụ 2-6, kiểm tra từng bộ giá trị.
  - *Tất nhiên có thể làm theo cách khác.*

Bài tập 2.8. Một số lẻ

- Viết chương trình nhập vào một dãy n số nguyên.
- Nhập vào một số k < n.
- Liệt kê tất cả các bộ k số trong dãy n số nguyên được vào có chứa đúng 1 số lẻ.
- Chương trình in ra NO nếu không có bộ giá trị thỏa mãn.

- Gợi ý:
  - Có thể phát triển ví dụ 2-6, kiểm tra từng bộ giá trị.
  - *Tất nhiên có thể làm theo cách khác.*

Bài toán liệt kê hoán vị

- Liệt kê các hoán vị của n số 0, 1, …, *n - 1*
- Cấu trúc phát triển lời giải bài toán:
  - Biểu diễn hoán vị bằng mảng x gồm n phần tử.
  - Sử dụng một mảng đánh dấu m để hỗ trợ chọn giá trị cho các vị trí, m[v] = 1 nếu v đã được chọn; m[v] = 0 nếu ngược lại (có thể chọn v).
  - Để chọn giá trị cho x[i] chúng ta duyệt tất cả các giá trị v từ 0 tới n -1. Nếu m[v] == 1 thì bỏ qua; Nếu ngược lại thì gán x[i] = v và đánh dấu m[v] = 1. Sau khi duyệt giá trị v thì hủy đánh dấu, gán m[v] = 0.
  - Sau khi chọn giá trị cho x[i], nếu i == n - 1 thì xuất hoán vị, nếu ngược lại tiếp tục gọi đệ quy.

Ví dụ 2.7. Liệt kê các hoán vị của n giá trị

vd2-7.c

```c
#include <stdio.h>

#define N 100
int x[N];
char m[N] = {0};
int n;

void output() {
  for (int i = 0; i < n; ++i) {
    printf("%d ", x[i]);
  }
  printf("\n");
}
void solve(int i) {
  for (int v = 0; v < n; ++v) {
    if (m[v]) {
      continue;
    }
    x[i] = v;
    m[v] = 1;
    if (i == n - 1) {
      output();
    } else {
      solve(i + 1);
    }
    m[v] = 0;
  }
}

int main() {
  printf("Nhap n: ");
  scanf("%d", &n);
  solve(0);
  return 0;
}
```

Bài tập 2.9. Cực đại địa phương

- Viết chương trình nhập một số nguyên n > 3 và một dãy n số nguyên.
- Một giá trị trong dãy được gọi là cực đại địa phương nếu nó lớn hơn phần tử đứng bên trái và phần tử đứng bên phải.
- Chương trình cần tìm và in ra hoán vị của dãy số ban đầu có số lượng cực đại địa phương là lớn nhất. Có thể in ra hoán vị bất kỳ trong trường hợp có nhiều đáp án.
- Ví dụ: n = 5, dãy 1 2 3 4 5 => #cực đại địa phương = 0
- Dãy 1 **3** 2 **5** 4 => #cực đại địa phương = 2

Gợi ý: Trong phạm vi bài tập này có thể phát triển ví dụ 2.7, đếm số cực đại địa phương cho mỗi hoán vị (vét cạn).

Bài tập 2.10. Hoán vị chẵn lẻ

- Viết chương trình nhập một số nguyên n > 1 và một dãy n số nguyên khác nhau.
- Chương trình in ra tất cả các hoán vị thỏa mãn điều kiện xen kẽ số chẵn và số lẻ. Chương trình in ra NO nếu không tồn tại hoán vị nào như vậy.
- Ví dụ: n = 3, dãy 1 2 3, các kết quả là:
  - 1 2 3
  - 3 2 1

Gợi ý: Trong phạm vi bài tập này có thể phát triển ví dụ 2.7, kiểm tra điều kiện xen kẽ cho tất cả các hoán vị (vét cạn).

Bài tập 2.11. Bài toán xếp quân hậu

- Viết chương trình xếp n quân hậu trên một bàn cờ tương tự bàn cờ vua (cờ quốc tế) sao cho không có 2 quân hậu nào ăn được nhau
- Ví dụ lời giải với n = 4

	0	1	2	3
0		X		
1				X
2	X			
3			X	

Bài tập 2.12. Các nghiệm nguyên dương

- Viết chương trình nhập n và M là các số nguyên dương thỏa mãn điều kiện n <= M,
- Liệt kê tất cả các nghiệm nguyên dương của phương trình:

$$x_0 + x_1 + \ldots + x_{n-1} = M$$

Bài tập 2.13. Sudoku

- Điền các chữ số từ 1 đến 9 vào các ô trong bảng vuông 9x9 sao cho trên mỗi hàng, mỗi cột và mỗi bảng vuông con 3x3 đều có mặt đầy đủ các chữ số từ 1 đến 9

1	2	3	4	5	6	7	8	9
4	5	6	7	8	9	1	2	3
7	8	9	1	2	3	4	5	6
2	1	4	3	6	5	8	9	7
3	6	5	8	9	7	2	1	4
8	9	7	2	1	4	3	6	5
5	3	1	6	4	2	9	7	8
6	4	2	9	7	8	5	3	1
9	7	8	5	3	1	6	4	2

ĐẠI HỌC BÁCH KHOA

25 YEARS ANNIVERSARY SOICT

VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

Thank you for your attentions!

soict.hust.edu.vn/ fb.com/groups/soict