TRƯỜNG THCS HAI BÀ TRƯNG

CHÀO MỪNG QUÝ THẦY CÔ, CÁC EM HỌC SINH TRONG ĐỘI TUYỂN CÁC TRƯỜNG THAM DỰ CHUYÊN ĐỀ MÔN TIN HỌC

Quận 3, ngày 15 tháng 11 năm 2022

GV thực hiện: Phạm Ngọc Quang

NỘI DUNG

1. Kính mời Quý Thầy Cô cùng các em học sinh tham gia trải nghiệm bài toán Tin học
2. Góp ý kiến, thảo luận giải pháp CT. BDHSG bộ môn
3. Chuẩn bị chuyên đề HK2, ma trận đề thi HK1-K6, 7.

ĐẶT VẤN ĐỀ

MỘT CÂY KIM BẰNG VÀNG RƠI VÀO ĐỐNG RƠM?

- Em hãy nêu các giải pháp để tìm được cây kim bằng vàng ?

1. GIẢI PHÁP ĐIỂN HÌNH

- Một nghệ sĩ của Ý đã dành 48 giờ đồng hồ lăn lộn trong đống rơm với mục đích tìm một... cây kim.

https://thanhnien.vn/mo-kim-trong-dong-rom-post432222.html

2. GIẢI PHÁP TỐN THỜI GIAN VÀ NHIỀU CHI PHÍ

- Cho con bò ăn hết đống rơm ?

MỤC TIÊU BÀI HỌC

- Học sinh biết vận dụng PP vét cạn khi giải bài toán Tin học
- Hiểu vai trò chức năng từng vòng lặp
- Biết cải tiến không gian nghiệm tìm kiếm:
  Chỉ số đầu lặp – chỉ số cuối lặp
- Vận dụng tối ưu cho từng bài toán Tin học

1 MÔ HÌNH TỔNG QUÁT GIẢI BẰNG PP - VÉT CẠN

C++

```
for( int i=1; i ≤ N; i ++)
{
    for( int j=1; i ≤ N; i ++)
    {
        for( int k=1; i ≤ N; i ++)
        {
            //Code ...
        }
    }
}
```

FreePascal

```
for i:=1 to n do
begin
    for j:=1 to  N do
    begin
        for k:=1 to N do
        begin
            //Code ...
        end;
    end;
end;
```

2 THỜI GIAN THỰC HIỆN CHƯƠNG TRÌNH

```
C++
for( int i=1; i ≤ N; i ++) {1}
{
    for( int j=1; i ≤ N; i ++) {2}
    {
        for( int k=1; i ≤ N; i ++) {3}
        {
            //Code ...
        }
    }
}
```

Gọi O lớn là thời gian thực hiện 1 vòng lặp:

- Thời gian thực hiện ở mỗi vòng lặp {1}, {2}, {3} là O(N). Vì 3 vòng lặp lồng nhau, nên ta có:

  O(N).O(N).O(N) = O(N.N.N)= O(N^3)

- Độ phức tạp thuật toán là O(N^3)
- PP vét cạn không khả thi với N ≤ 10^4.

3 GIẢI BÀI TOÁN TIN BẰNG PHƯƠNG PHÁP VÉT CẠN

Ưu điểm: 

Vét cạn là xét hết mọi trường hợp nghiệm, có khả năng hoặc không. Trong lập trình, đây là phương pháp tiếp cận bài toán dễ nhất khi ta chưa tìm được giải pháp nào hiệu quả hơn.

Khuyết điểm: 

- Tốn nhiều thời gian
- Chương trình chạy chậm
- Chỉ đạt một số Test nhỏ
- ...
- Không tối ưu.

4 BÀI TOÁN

Tên bài DAYCON.cpp (C++) hoặc DAYCON.pas (FreePascal)
Viết chương trình đọc vào dãy số nguyên $a_1, a_2 \ldots a_N$ có N phần tử $(1 \leq N \leq 10^4)$. Yêu cầu xuất ra các dãy con liên tiếp.

DAYCON.INP	DAYCON.OUT
4 2 5 4 3	2 5 4 3 2 5 5 4 4 3 2 5 4 5 4 3 2 5 4 3

Phân tích:
Dãy con được lấy từ dãy mẹ ban đầu. Mỗi dãy con phải được viết xuống dòng, tổng cộng có 10 dãy con được xác định.

Code 0

```
void xuatDayCon_0()
{
    int a[10005], n;
    cin >> n;
    for (int i=1; i<=n;i++) cin>> a[i];
    // Áp dụng mô hình vét cạn
    for (int i= 1;i<=n;i++)          // 1
    {
        for(int j=1;j<=n;j++)        // 2
        {
            for (int k=1;k<=n;k++)   // 3
            {
                cout << a[k] <<" ";
            }
            cout <<'\n';
        }
    }
}
```

KẾT QUẢ

DAYCON.OUT

```
1.  [2][5][4][3]
2.    2 5 4 3
3.    2 5 4 3
4.    2 5 4 3
5.    2 5 4 3
6.    2 5 4 3
7.    2 5 4 3
8.    2 5 4 3
9.    2 5 4 3
10.   2 5 4 3
11.   2 5 4 3
12.   2 5 4 3
13.   2 5 4 3
14.   2 5 4 3
15.   2 5 4 3
16.   2 5 4 3
```

i=1 | i=2 | i=3 | i=4

[1] [2] [3] [4]

2 5 4 3

j=1 k=1 k=2 k=3 k=4=n → xuống dòng

j=2 k=1 k=2 k=3 k=4=n → xuống dòng

j=3 k=1 k=2 k=3 k=4=n → xuống dòng

j=4 k=1 k=2 k=3 k=4=n → xuống dòng

Nhận xét 0:

- Vòng for i: mỗi lượt của i tạo ra 4 dãy con. Cứ mỗi lượt của for j lại có for k=1→duyệt đến n, đồng thời xuất từng phần tử đến khi k=n mới kết thúc lượt của j. Kết quả là có 1 dãy con gồm nhiều phần tử, sau đó thì xuống dòng, kết thúc lượt j trước đó → Nên cấu hình bị trùng ở mỗi lượt i và j tiếp theo.
- Cần cải tiến như thế nào ?

Code 1

```
void xuatDayCon_1()
{
    int a[100005], n;
    cin >> n;
    for (int i=1; i<=n;i++) cin>> a[i];
    // Áp dụng mô hình vét cạn
    for (int i= 1;i<=n;i++)          // 1
    {
        for(int j=1;j<=n;j++)        // 2
        {
            for (int k=1;k<=n;k++)   // 3
            {
                cout << a[k] <<" ";
            }
            cout <<'\n';
        }
    }
}
```

KẾT QUẢ

DAYCON.OUT

```
1.  2 5 4 3
2.  2 5 4 3
3.  2 5 4 3
4.  2 5 4 3
5.  2 5 4 3
6.  2 5 4 3
7.  2 5 4 3
8.  2 5 4 3
9.  2 5 4 3
10. 2 5 4 3
11. 2 5 4 3
12. 2 5 4 3
13. 2 5 4 3
14. 2 5 4 3
15. 2 5 4 3
16. 2 5 4 3
```

i=1 (dòng 1–4), i=2 (dòng 5–8), i=3 (dòng 9–12), i=4 (dòng 13–16)

[1]	[2]	[3]	[4]
2	**5**	**4**	**3**

j=1 k=1 k=2 k=3 k=4=n → xuống dòng
j=2 k=1 k=2 k=3 k=4=n → xuống dòng
j=3 k=1 k=2 k=3 k=4=n → xuống dòng
j=4 k=1 k=2 k=3 k=4=n → xuống dòng

Cải tiến 1:

Nhận thấy cứ mỗi lượt của j thì k luôn bắt đầu từ k=1 → duyệt đến n. Vì :

Vị trí truy xuất của i, j, k trong mảng a

j=1
i=1 n

[1]	[2]	[3]	[4]	… [10005]
2	**5**	**4**	**3**	

k=1

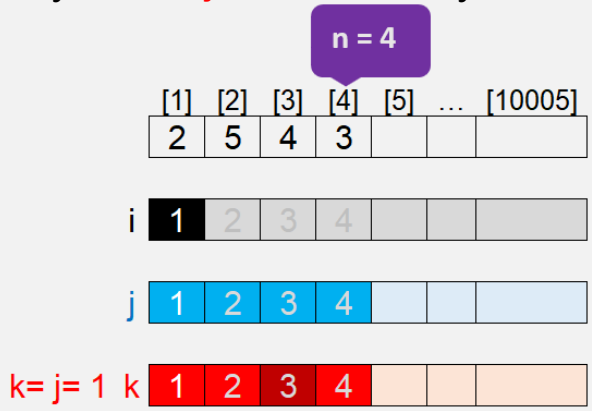
Đến đây ta vẫn chưa thấy rỏ vai trò của for k chỉ thấy đơn thuần duyệt từ k=1 và xuất giá trị a[k] cho đến hết n;

Cải tiến vòng lặp 3

Code 1

```
void xuatDayCon_1()
{
    int a[100005], n;
    cin >> n;
    for (int i=1; i<=n;i++) cin>> a[i];
    // Áp dụng mô hình vét cạn
    for (int i= 1;i<=n;i++)                    // 1
    {
        for(int j=1;j<=n;j++)                  // 2
        {
            for (int k=j; k<= ?      ;k++)     //3
            {
                cout << a[k] <<" ";
            }
            cout <<'\n';
        }
    }
}
```

- Tạm thời xem for i = 1 → duyệt đến n;
- for j=1 → duyệt đến n;
- Ở mỗi lượt thứ j được xem là vị trí đầu dãy con;
- Vậy for k= j sẽ được duyệt trong đoạn [j .. X?]



Tại thời điểm i=1, j=1,k=j= 1 ta cần lấy nghiệm 2 xuống dòng
Tại thời điểm i=1, j=2,k=j= 2 ta cần lấy nghiệm 5 xuống dòng
Tại thời điểm i=1, j=3,k=j= 3 ta cần lấy nghiệm 4 xuống dòng
Tại thời điểm i=1, j=4,k=j= 4 ta cần lấy nghiệm 3 xuống dòng

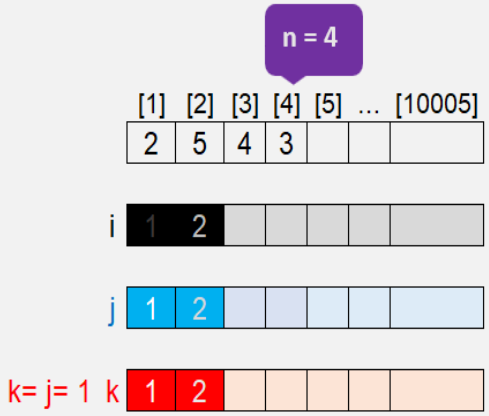
Vậy khả năng k=j duyệt đến k<=j ?

Cải tiến vòng lặp 3

Code 1

```
void xuatDayCon_1()
{
    int a[100005], n;
    cin >> n;
    for (int i=1; i<=n;i++) cin>> a[i];
    // Áp dụng mô hình vét cạn
    for (int i= 1;i<=n;i++)                    // 1
    {
        for(int j=1;j<=n;j++)                  // 2
        {
            for (int k=j; k<= ?      ;k++)     //3
            {
                cout << a[k] <<" ";
            }
            cout <<'\n';
        }
    }
}
```

- Tạm thời xem for i = 1 → duyệt đến n;
- for j=1 → duyệt đến n;
- Ở mỗi lượt thứ j được xem là vị trí đầu dãy con;
- Vậy for k= j sẽ được duyệt trong đoạn [j .. X?]



Tại thời điểm i=2, j=1,k=j= 1 ta cần lấy nghiệm **2** 5 xuống dòng vì k<=j

Ta cần 2 và 5 nhưng lấy được nghiệm 2 thì đã xuống dòng vì k<=j=1

Vì i=2 nên j=2, k =j= 2 nên nghiệm tiếp theo là **5** 4 xuống dòng vì k<=j

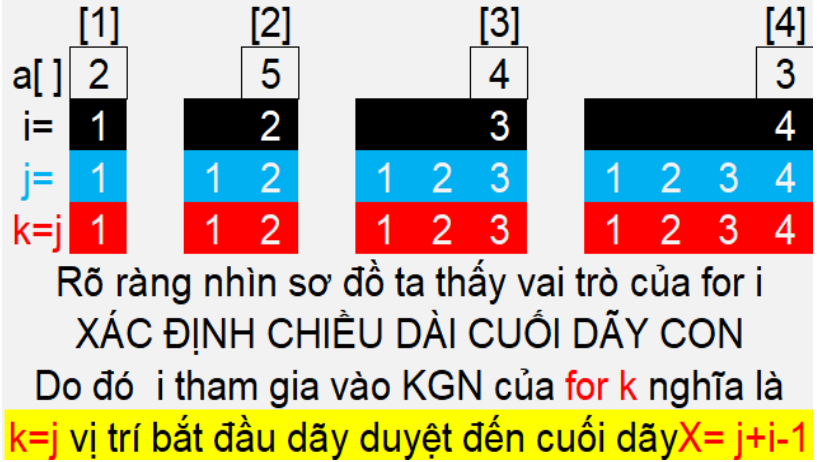
Kết luận khả năng k=j duyệt đến k<=j ? Không thành công ! Do k<=j

Cải tiến vòng lặp 3

Code 1

```
void xuatDayCon_1()
{
    int a[100005], n;
    cin >> n;
    for (int i=1; i<=n;i++) cin>> a[i];
    // Áp dụng mô hình vét cạn
    for (int i= 1;i<=n;i++)                    // 1
    {
        for(int j=1;j<=n;j++)                  // 2
        {
            for (int k=j; k<= ?      ;k++)     //3
            {
                cout << a[k] <<" ";
            }
            cout <<'\n';
        }
    }
}
```

Đến đây ta thấy rõ vai trò của for i=1→n;
Khi i=1→ 4 dãy con {1 phần tử}
Khi i=2→ 3 dãy con {2 phần tử}
… do vậy **for i** tham gia qua trình tìm dãy con ở vòng lặp for k. Cụ thể sơ đồ bên dưới:
- Lượt i=2 thì mới tạo được đoạn nghiệm cho for j=1..2;
- Lượt i=3 thì mới tạo được đoạn nghiệm for j=1 .. 3
Quan sát sơ đồ bên dưới:



Code 1

```
void xuatDayCon_1()
{
    int a[100005], n;
    cin >> n;
    for (int i=1; i<=n;i++) cin>> a[i];
    // Áp dụng mô hình vét cạn
    for (int i= 1;i<=n;i++)                 // 1
    {
        for(int j=1;j<=n;j++)               // 2
        {
            for (int k=j;k<=j+k-1;k++) // 3
            {
                cout << a[k] <<" ";
            }
            cout <<'\n';
        }
    }
}
```

KẾT QUẢ

DAYCON.OUT

[1] [2] [3] [4]

2 5 4 3

1. 2
2. 5
3. 4
4. 3

i=1

j=1 k= j = 1 → k <= j+i - 1= 1+1-1= 1→ xuống dòng

j=2 k= j = 2 → k <= j+i - 1= 1+2-1= 2→ xuống dòng

j=3 k= j = 3 → k <= j+i - 1= 1+3-1= 3→ xuống dòng

j=4 k= j = 4 → k <= j+j - 1= 1+4-1= 4→ xuống dòng

5. 2 5
6. 5 4
7. 4 3
8. 3 0

i=2

Hết lượt i=1, đến lượt i=2, các bước chạy tương tự như trên. Nhưng đến cuối lượt j ?

←Cuối lượt j= 4 k=4 → k<=j+i-1 = 2+ 4 - 1 = 5 → Dư.

9. 2 5 4
10. 5 4 3
11. 4 3 0
12. 3 0 0

i=3

13. 2 5 4 3
14. 5 4 3 0
15. 4 3 0 0
16. 3 0 0 0

i=4

Code 2

```
void xuatDayCon_2()
{
    int a[100005], n;
    cin >> n;
    for (int i=1; i<=n;i++) cin>> a[i];
    // Áp dụng mô hình vét cạn
    for (int i= 1;i<=n;i++)                  // 1
    {
        for(int j=1;j <= n-i+1;j++)      // 2
        {
            for (int k=j;k<=j+k-1;k++) // 3
            {
                cout << a[k] <<" ";
            }
            cout <<'\n';
        }
    }
}
```

KẾT QUẢ Cải tiến vòng lặp 2

DAYCON.OUT

```
1.  2
2.  5
3.  4
4.  3
5.  2 5
6.  5 4
7.  4 3
8.  3 0
9.  2 5 4
10. 5 4 3
11. 4 3 0
12. 3 0 0
13. 2 5 4 3
14. 5 4 3 0
15. 4 3 0 0
16. 3 0 0 0
```

i=1 (dòng 1–4)

i=2 (dòng 5–8)

i=3 (dòng 9–12)

i=4 (dòng 13–16)

- Lúc này for j được xem như là vị trí bắt đầu của dãy con for j=1; Vậy KGN j duyệt đến đâu ?

- Có i đóng vai trò xác định độ dài của dãy con nên i được xem là vị trí cuối dãy của dãy con duyệt theo n. Nên số lượng phần tử cần duyệt của j ở mỗi lượt của for i sẽ là : n - i + 1;

Code 2

```
void xuatDayCon_2()
{
    int a[100005], n;
    cin >> n;
    for (int i=1; i<=n;i++) cin>> a[i];
    // Áp dụng mô hình vét cạn
    for (int i= 1;i<=n;i++)                  // 1
    {
        for(int j=1; j<= n-i+1 ;j++)      // 2
        {
            for (int k=j;k<=j+k-1;k++) // 3
            {
                cout << a[k] <<" ";
            }
            cout <<'\n';
        }
    }
}
```

KẾT QUẢ

DAYCON.OUT

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

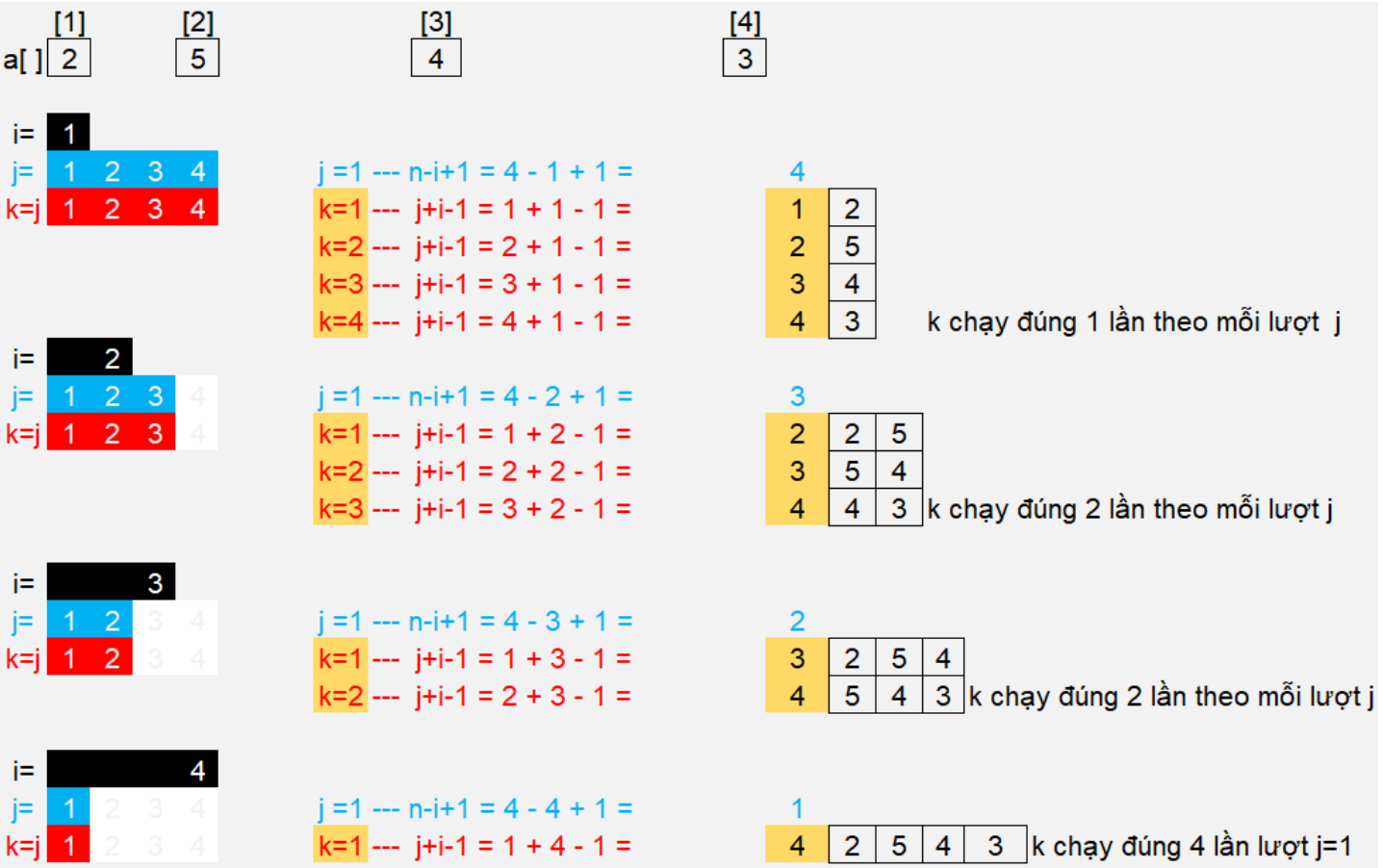
TỔNG QUÁT

Vòng for 1: Cho biết độ dài của dãy con có i phần tử;

Vòng for 2: Cho biết dãy con bắt đầu từ vị trí j, vậy vị trí bắt đầu của dãy con cuối cùng có độ dài i sẽ là n-i+1;

Vòng for 3: Dùng để duyệt qua các phần tử của dãy con bắt đầu từ vị trí đầu là j đến vị trí cuối là i+j-1 để xử lý.

KL: Gần như tất cả các bài toán Tin học đều có thể giải quyết bằng PP vét cạn→ Nhưng không tối ưu; Do đó HS cần phải nắm vững, hiểu rõ vai trò của vòng lặp, biết cải tiến KGN tìm kiếm từ 3 for → 2 for → 1 for bằng nhiều kỹ thuật, thông qua rèn luyện nhiều bài tập dựa vào kinh nghiệm để đạt kết quả tốt nhất.


[1] [2] [3] [4]
a[ ] 2 5 4 3
i= 1
j= 1 2 3 4
k=j 1 2 3 4
j =1 --- n-i+1 = 4 - 1 + 1 = 4
k=1 --- j+i-1 = 1 + 1 - 1 =
k=2 --- j+i-1 = 2 + 1 - 1 =
k=3 --- j+i-1 = 3 + 1 - 1 =
k=4 --- j+i-1 = 4 + 1 - 1 =
1 2
2 5
3 4
4 3
k chạy đúng 1 lần theo mỗi lượt j
i= 2
j= 1 2 3 4
k=j 1 2 3 4
j =1 --- n-i+1 = 4 - 2 + 1 = 3
k=1 --- j+i-1 = 1 + 2 - 1 =
k=2 --- j+i-1 = 2 + 2 - 1 =
k=3 --- j+i-1 = 3 + 2 - 1 =
2 2 5
3 5 4
4 4 3
k chạy đúng 2 lần theo mỗi lượt j
i= 3
j= 1 2 3 4
k=j 1 2 3 4
j =1 --- n-i+1 = 4 - 3 + 1 = 2
k=1 --- j+i-1 = 1 + 3 - 1 =
k=2 --- j+i-1 = 2 + 3 - 1 =
3 2 5 4
4 5 4 3
k chạy đúng 2 lần theo mỗi lượt j
i= 4
j= 1 2 3 4
k=j 1 2 3 4
j =1 --- n-i+1 = 4 - 4 + 1 = 1
k=1 --- j+i-1 = 1 + 4 - 1 =
4 2 5 4 3
k chạy đúng 4 lần lượt j=1

BÀI TẬP VẬN DỤNG

Tên bài SUMK.cpp (C++) hoặc SUMK.pas (FreePascal)

Viết chương trình đọc vào dãy số nguyên $a_1, a_2 \ldots a_N$ có N phần tử số K ($1 \leq K \leq N \leq 10^4$). Yêu cầu xuất ra tổng các dãy con có đúng K phần tử liên tiếp.

SUMK.INP	SUMK.OUT
4 3 2 5 4 3	11 12

Yêu cầu:

1. Giải bằng PP vét cạn
2. Tối ưu còn 2 for
3. Tối ưu còn 1 for áp dụng
   - Kỹ thuật tính tổng trước
4. Kỹ thuật duyệt 2 con trỏ
5. Tìm kiếm nhị phân
6. Kỹ thuật quay lui
7. Qui hoạch động

NỘI DUNG

1. Kính mời Quý Thầy Cô cùng các em học sinh tham gia trải nghiệm bài toán Tin học

2. Góp ý kiến, thảo luận giải pháp CT. BDHSG bộ môn

3. Chuẩn bị chuyên đề HK2, ma trận đề thi HK1-K6, 7.