BÀI 7. PREFIX & SUFFIX

NỘI DUNG BÀI HỌC

- Bài toán tổng trước (prefix sum)
- Bài toán tính giá trị đa thức.
- Bài toán tổng con lớn nhất trên mảng

1. BÀI TOÁN PREFIX SUM

BÀI TOÁN PREFIX, SUFFIX

- Phát biểu: Cho dãy n phần tử A[1..n]
  - P[i] được gọi là tổng trước thứ i trên dãy A nếu P[i] = $\sum$ A[j] với j = 1..i
  - S[i] được gọi là tổng sau thứ i trên dãy A, nếu S[i] = $\sum$ A[j] với j = i..n;
- Bài toán thiết kế thuật toán trên PRAM:
  - INPUT: A[1..n]
  - OUTPUT: P[1..n] hoặc S[1..n]
- Hai bài toán này về bản chất là như nhau

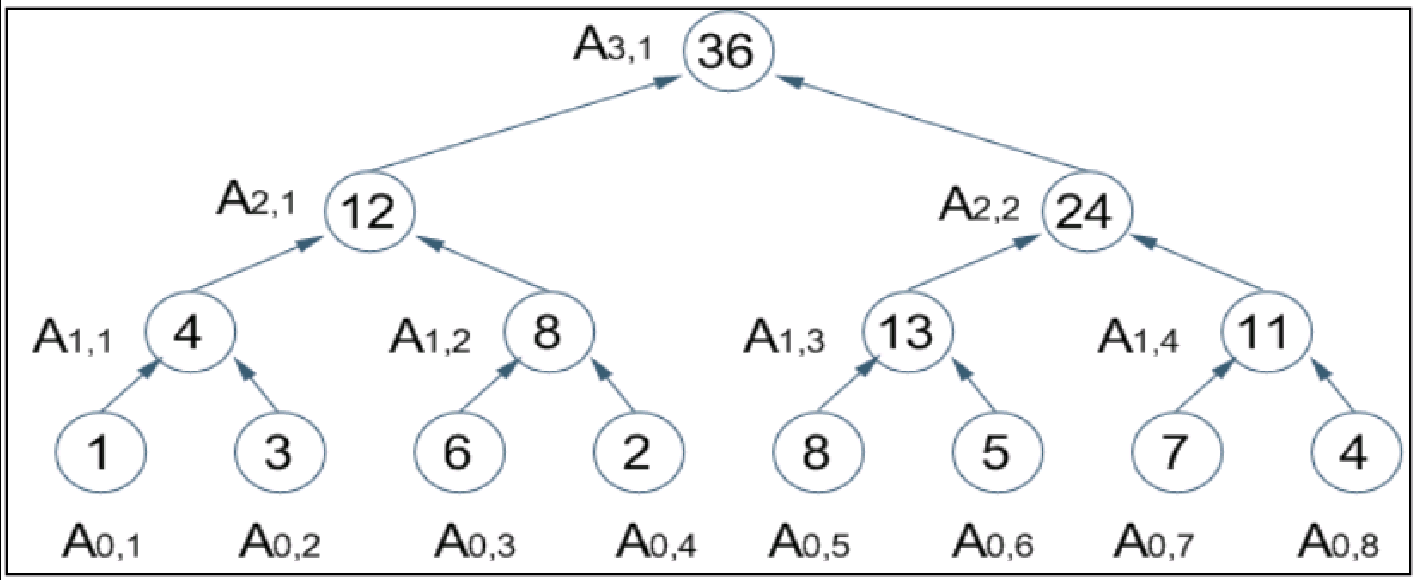
BÀI TOÁN PREFIX SUM

- Các cách tiếp cận:
  - Cách 1: Áp dụng kỹ thuật cây cân bằng và phát triển nhân đôi.
  - Cách 2: Phương pháp đệ quy.
  - Cách 3: Áp dụng kỹ thuật kiểu con trỏ nhảy.

CÁCH 1 - KỸ THUẬT CÂY CÂN BẰNG

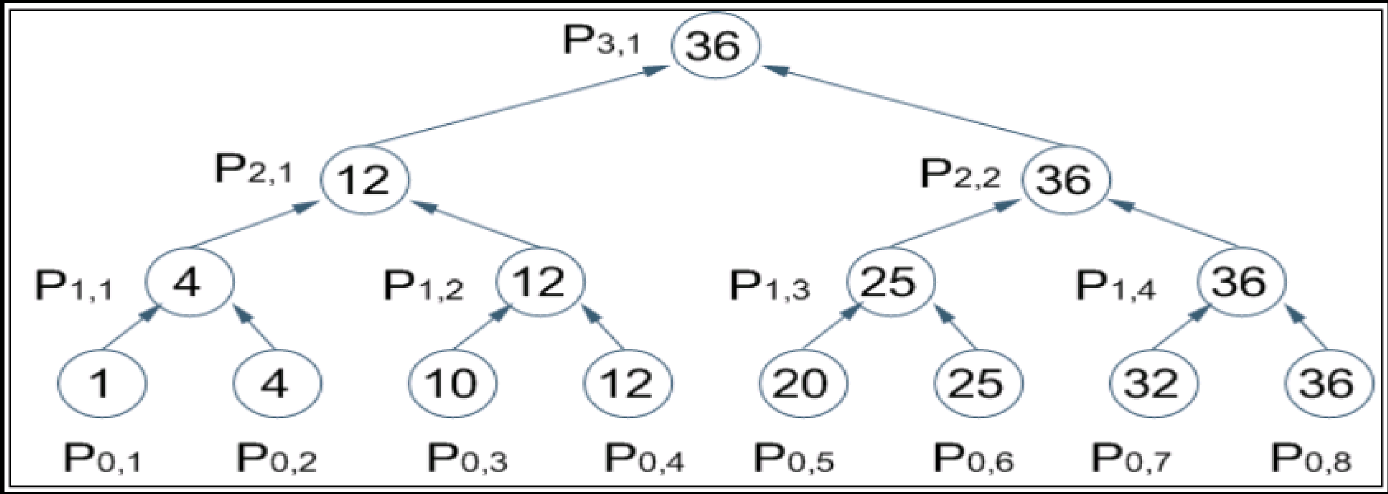
- Ý tưởng:
  - Xây dựng cây cân bằng
  - Xây dựng cây P, mỗi nút có tên là $P_{i,j}$ với i là chỉ số mức, j là chỉ số BXL
  - Giả sử $P_{i,j}$ là gốc của 1 cây con có lá ngoài cùng là $P_{0,k}$ thì giá trị $P_{0,k} = \sum A[t]$ với t=1..k

B1.XÂY DỰNG CÂY CÂN BẰNG



- Mỗi nút trên cây được biểu diễn bởi $A_{i,j}$ trong đó:
  - i là chỉ số mức
  - j là chỉ số BXL
- Thuật toán không đổi $A_{i+1,j} = A_{i,2j-1} + A_{i,2j}$

B2.XÂY DỰNG CÂY P



- Một số nhận xét:
  - Cây P xây dựng từ trên xuống tới đỉnh là $P_{k,1} = A_{k,1} = \sum A[i]$ với i=1..n, k=logn
  - $P_{i,1} = A_{i,1}$ với i=k-1..0
  - $P_{i,j} = P_{i+1,j/2}$ với j chẵn; $P_{i,j} = P_{i+1,[j/2]}$ + Ai,j với j lẻ, i= k-1 … 0

KỸ THUẬT CÂY CÂN BẰNG

```
input       : A[1..n]; n = 2^k
output      : P[1..n] | P[i] = Prefix_sum[i]
begin
    for i = 1 to n do in parallel
        A[0,i]     =     A[i];
    end parallel
    for i = 1 to k do
        for j = 1 to n/2^i do in parallel
            A[i,j]=     A[i-1,2j-1] + A[i-1,2j];
        end parallel
    end for
    P[k,1]     =     A[k,1];
    for i = k -1 downto 0 do
        for j = 1 to 2^(k - i) do in parallel
            if j = 1 then P[i,1] = A[i,1];
            else if j chẵn then P[i,j] = P[i+1,j/2]
            else P[i,j] = P[i+1,[j/2]] + A[i,j]
        end parallel
    end for
    for i = 1 to n do in parallel
        P[i]  =     P[0,i];
    end parallel
end.
```

ĐÁNH GIÁ ĐỘ PHỨC TẠP

- Thuật toán chia thành 2 phần:
  - Phần 1: Xây dựng cây cân bằng O(logn)
  - Phần 2: Xây dựng cây P với O(logn) bước tuần tự
    - Xét các phép toán gán khi xây dựng cây P. Các nút Pi,j với j chẵn và lẻ cùng đọc giá trị $P_{i+1,[j/2]}$.
    - Với j chẵn: $P_{i,j}$ đọc $P_{i+1,j/2}$
    - Với j lẻ: $P_{i,j}$ đọc $A_{i,j}$ trước, sau đó đọc $P_{i+1,[j/2]}$.

CÁCH 2 - GIẢI THUẬT ĐỆ QUY

- Thuật toán cây cân bằng xây dựng đệ quy:

```
function S = Reduce(A[1..n])
begin
        if n = 1 then
                S = A[1];
                return S;
        end if
        for i = 1 to n/2 do in parallel
                A[i] = A[2i-1] + A[2i]
        end parallel
        S = Reduce(A[1..n/2];
end
```

GIẢI THUẬT ĐỆ QUY

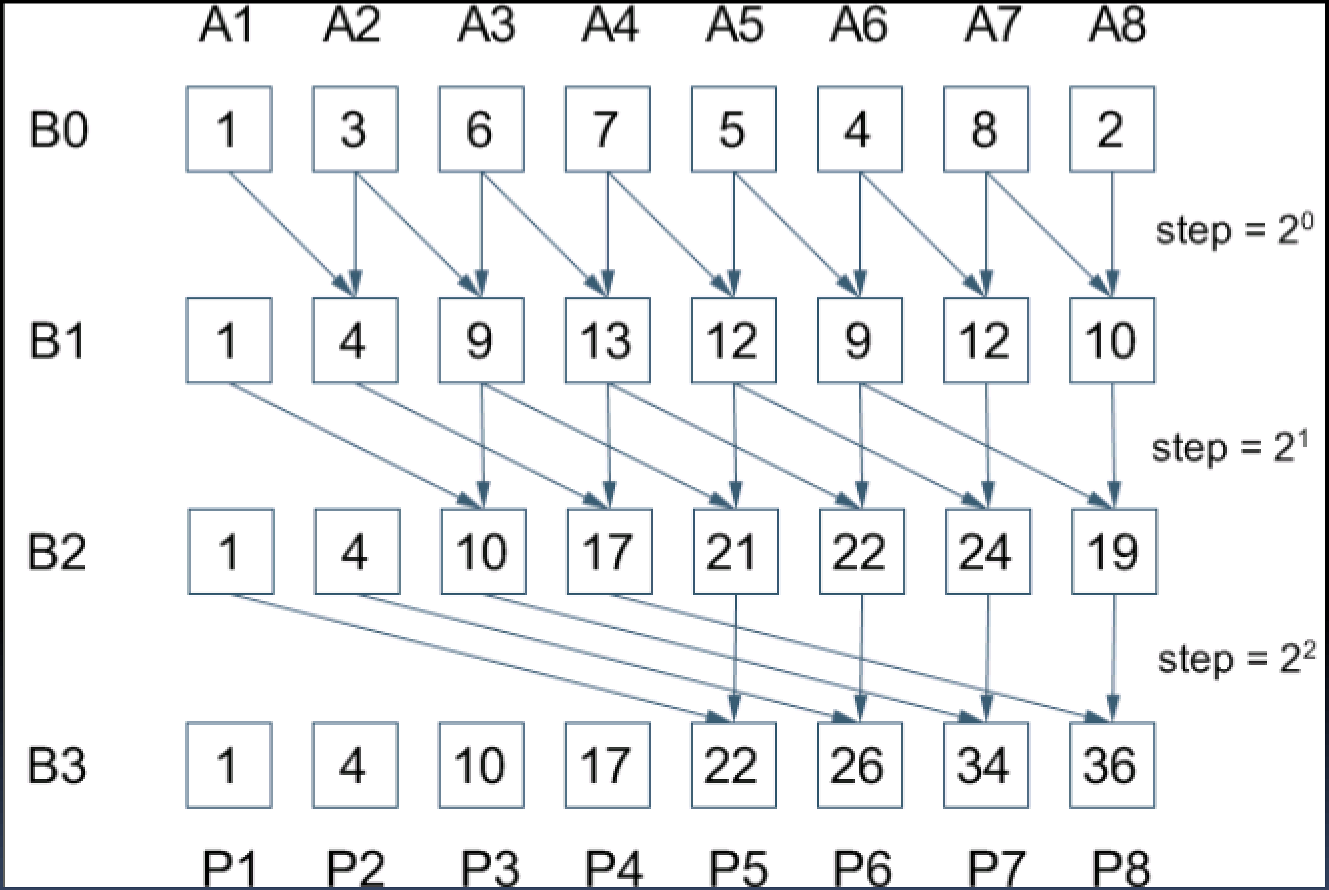
- 1 lần gọi đệ quy O(1)
- Thời gian tổng: $O(\log_2 n)$

```
function P[1..n] = Scan(X[1..n])
begin
    if n = 1 then
        P[1] = X[1];
        return P;
    end if
    for i = 1 to n/2 do in parallel
        Y[i] = X[2i-1] + X[2i]
    end parallel

    Z[1..n/2] = Scan(Y[1..n/2]);

    for i = 1 to n do in parallel
        if i chẵn then  P[i] = Z[i/2];
        elseif i = 1 then P[1] = X[1];
        else P[i] = Z[(i-1)/2] + X[i];

    end parallel
    return P;
end
```

CÁCH 3 - GIẢI THUẬT KIỂU CON TRỎ NHẢY

- Ý tưởng:
  - Ban đầu khởi tạo: $P_0[i] = A[i]$
  - Tại bước thứ k:
    - Step = 2^{k-1}
    - $P_k[i] = P_{k-1}[i] + P_{k-1}[i\text{-step}]$ với mọi i sao cho **step <i <=n**

MINH HỌA



A1 A2 A3 A4 A5 A6 A7 A8
B0 1 3 6 7 5 4 8 2
step = 2^0
B1 1 4 9 13 12 9 12 10
step = 2^1
B2 1 4 10 17 21 22 24 19
step = 2^2
B3 1 4 10 17 22 26 34 36
P1 P2 P3 P4 P5 P6 P7 P8

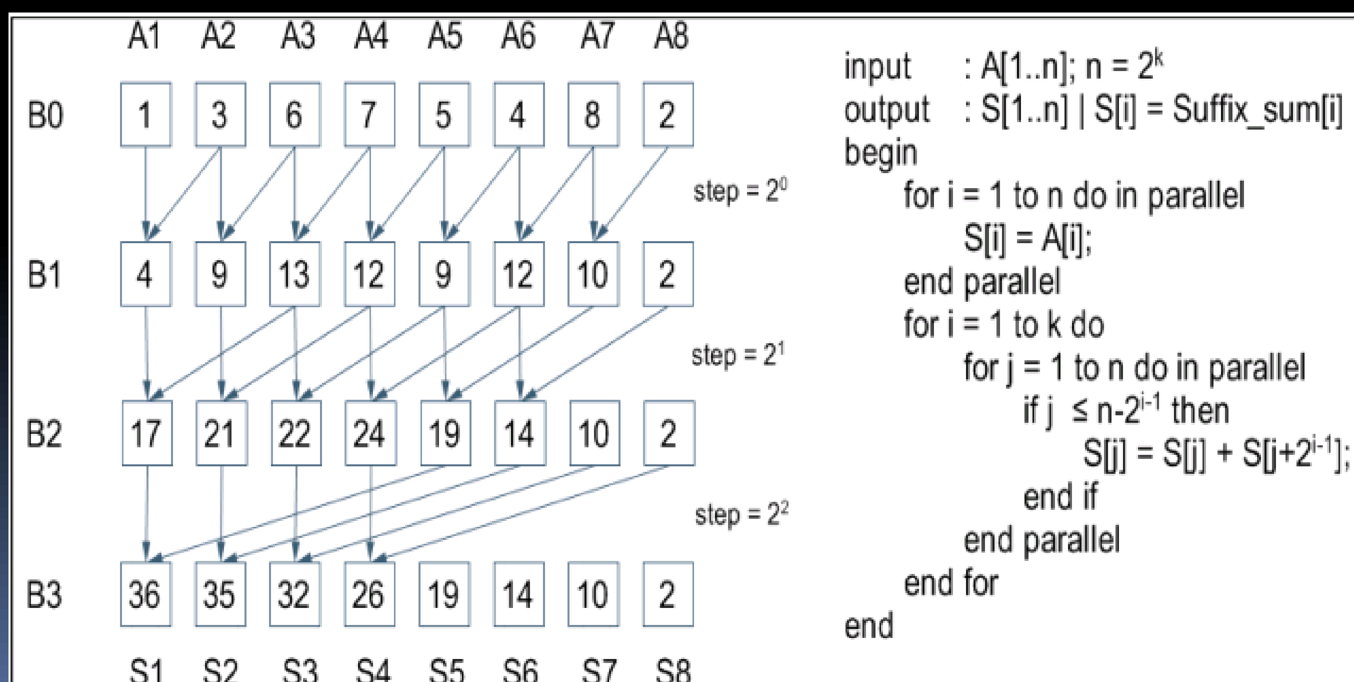
THUẬT TOÁN MINH HỌA

- Thời gian thực hiện: Số bước lặp tuần tự O(k) = O(logn)

```
input     : A[1..n]; n = 2^k
output    : P[1..n] | P[i] = Prefix_sum[i]
begin
    for i = 1 to n do in parallel
        P[i] = A[i];
    end parallel
    for i = 1 to k do
        for j = 1 to n do in parallel
            if j > 2^(i-1) then
                P[j] = P[j] + P[j-2^(i-1)];
            end if
        end parallel
    end for
end
```

BÀI TOÁN SUFFIX SUM

- Bài toán Suffix tương tự như Prefix
- Trong 3 cách trên, cách thứ 3 thường được viết vì tính đơn giản do đó viết Suffix với cách 3



2. BÀI TOÁN ĐA THỨC

PHÁT BIỂU BÀI TOÁN

- Giả sử các hệ số A[0..n] của 1 đa thức bậc n cho trước. Hãy tính giá trị đa thức tại điểm x_0 bất kỳ.
- Input: $a_0, a_1, \ldots a_n$. PRAM n BXL
- Output: $P(x_0)$ với $P(x) = \sum a_i x^i$. i = 0..n
- Phương pháp giải:
  - Sử dụng kỹ thuật Prefix

ÁP DỤNG PREFIX

- Nếu các giá trị A[i] = x và thay phép toán cộng (+) bằng phép toán nhân (x) trong bài toán Prefix thì ta sẽ thu được kết quả là: $P[i] = x^i$.
- Sau khi thực hiện Prefix ta thực hiện tính tích vô hướng của 2 vector thông thường:
  - Vector 1: hệ số đa thức a[1..n]
  - Dãy Prefix tích: P[1..n]

THUẬT TOÁN

input : a[0..n] là các hệ số đa thức bận n: P(x); x_0 bất kỳ. ($n = 2^k$)

output : tính giá trị của P(x)

```
begin
    for i = 1 to n do in parallel
        X[i]  =   x0;                              O(1)
    end parallel
    for i = 1 to k do
        for j = 1 to n do in parallel
            if  j > 2^(i-1) then
                X[j] = X[j] + X[j-2^(i-1)];        O(log2n)
            end if;
        end parallel
    end for.
    for i = 1 to n do in parallel
        Y[i] = X[i]*a[i];                          O(1)
    end parallel
    for i = 1 to k do
        for j = 1 to n/2^i do in parallel
            Y[i] = Y[2i-1] + Y[2i];                O(log2n)
        end parallel
    end for;
    P(x0) = Y[1] + a[0];
end.
```

2. BÀI TOÁN TỔNG CON LỚN NHẤT

PHÁT BIỂU BÀI TOÁN

- Cho dãy giá trị A[1..n]. Hãy xác định đoạn con trong dãy sao cho tổng các phần tử của đoạn con là lớn nhất.
- Input: A[1..n]
- Output: (i,j) thuộc 1..n | {∑A[t] với t thuộc i..j} max.
- Ví dụ A[1..16]

3	2	-7	11	10	-6	4	9	-6	1	-2	-3	4	-3	0	2

- Mảng con lớn nhất (i,j) = (4,8). Sum = 28

GIẢI THUẬT TUẦN TỰ

- Độ phức tạp là $O(n^3)$

```
input    : A[1..n]
output   : (i,j) | { ∑A[t] với  i ≤ t ≤ j } max
begin
      max = - Inf                                   O(n³)
      for i = 1 to n do
            for j = 1 to n do
                  s = 0;
                  for k = i to j do
                        s = s + A[k];
                  end for
                  if s > max then
                        max = s;
                        save(i,j);
                  end if
            end for
      end for
end
```

GIẢI THUẬT TUẦN TỰ

- Cải tiến thuật toán với ý tưởng:

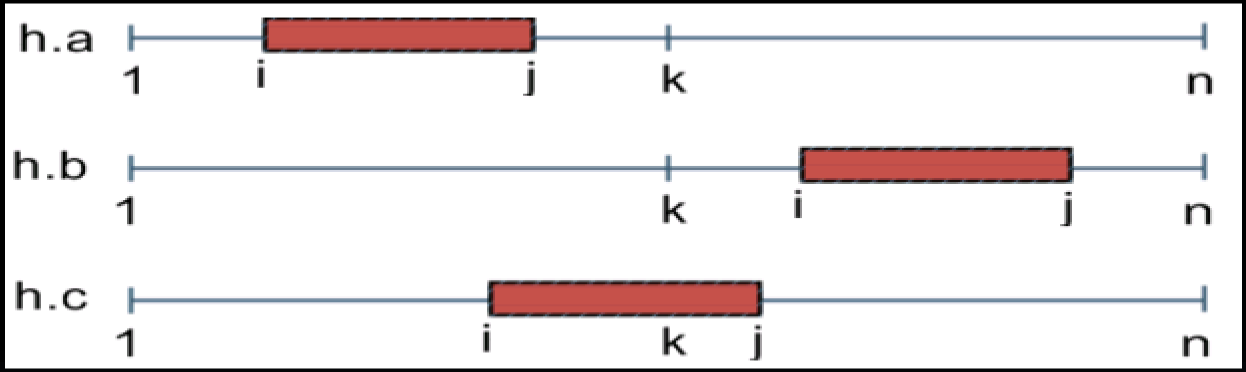
$$\sum_{k=i}^{j} a[k] = a[j] + \sum_{k=i}^{j-1} a[k]$$

```
input   : A[1..n]
output  : (i,j) | { ∑A[t] với  i ≤ t ≤ j } max
begin
        max = - Inf
        for i = 1 to n do
                s = 0;
                for j = i to n do
                        s = s + A[j];
                end for
                if s > max then
                        max = s;
                        save(i,j);
                end if
        end for
end
```

$O(n^2)$

GIẢI THUẬT TUẦN TỰ

- Phương pháp chia để trị



GIẢI THUẬT TUẦN TỰ

- Phương pháp chia để trị
- Thực hiện đệ quy với mảng A có cận bên trái là left, cận phải là right: S=(A, left, right)
  - Nếu left=right -> trả lại giá trị tại điểm đó.
  - Xét center là điểm giữa left, right:
    - Thực hiện đệ quy với S1=(A,left,center)
    - Thực hiện đệ quy với S2=(A,center,right)
    - Tính SM = suffix_summax(A, left, center);
    - Tính PM = prefix_summax(A, center, right);
    - Tính S3 = SM + PM – A[center]
    - Xác định S = max{S1, S2, S3}

GIẢI THUẬT TUẦN TỰ

```
function MaxSubVector(a, i, j);
begin
        if (i = j) then return a[i]
        else
        begin
                m:= (i+j)/2;
                wL:= MaxSubVector(a, i, m);
                wR:= MaxSubVector(a, m+1, j);
                wM:= MaxLeftVector(a, i, m)+ MaxRightVector(a, m+1, j);
                return max(wL, wR, wM);
        end;
end;
```

GIẢI THUẬT TUẦN TỰ

- Thời gian tính của các hàm MaxLeftVecto và MaxRightVector xác định với kích thước của mảng W(n) = n;
- Công thức đệ quy: T(n) = 2 * T(n/2) + W(n)
- Khi n = 1 thì T(1) = 1, thời gian thực hiện thuật toán sẽ là O(n*log2n)

GIẢI THUẬT TUẦN TỰ

```
function MaxLeftVector(a, i, j);
begin
    maxSum:= -Inf ; sum:=0;
    for k:= j downto i do
    begin
        sum:= sum+a[k];
        maxSum:= max(sum, maxSum);
    end;
    return maxSum;
end;
```

```
function MaxRightVector(a, i, j);
begin
    maxSum:= -Inf; sum:=0;
    for k:= i to j do
    begin
        sum:= sum+a[k];
        maxSum:= max(sum, maxSum);
    end;
    return maxSum;
end;
```

GIẢI THUẬT SONG SONG

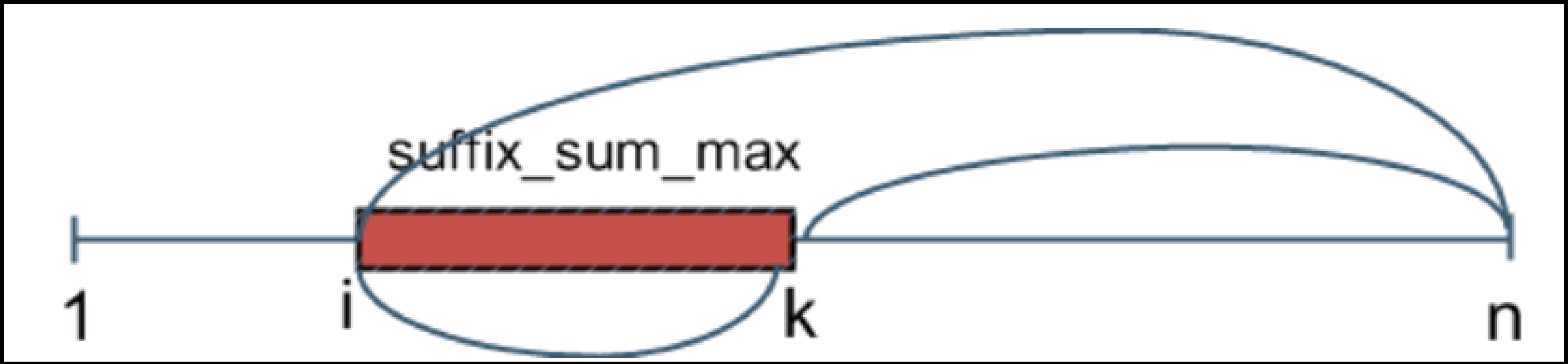
- Ý tưởng:
  - P_k xác định (i_k, j_k) có k:
    - Tính Suffix_sum_max đối với A[k];
    - Tính Prefix_sum_max đối với A[k];
- So sánh các tổng M_k -> giá trị lớn nhất
  - i, j đầu tiên & cuối cùng mà M_k lớn nhất

VÍ DỤ ĐOẠN CON LỚN NHẤT CHỨA A[7]

Q	3	2	-7	11	10	-6	4	9	-6	1	-2	-3	4	-3	0	2
$l(q_7)$	3	2	-7	11	10	-6	4									
$S_l(q_7)$	17	14	12	19	8	-2	4									
$r(q_7)$							4	9	-6	1	-2	-3	4	-3	0	2
$P_r(q_7)$							4	13	7	8	6	3	7	4	4	6

- M_s^7 = Maximum (S_l) = 19; M_p^7 = Maximum(P_r) = 13
- $M_s^7 + M_p^7 - q_7 = 19 + 13 - 4 = 28 = Max(q_7)$

XÁC ĐỊNH SUFFIX_SUM_MAX



- Nhận xét:
  - $S_{[i,k]} = S_{[i,n]} – S_{[k,n]}$ -> ta cần tính các suffix_sum của mảng A
  - $S_{[i,k]}$ max nếu $S_{[i,n]}$ max trong các $S_{[j,n]}$ với j thuộc 1..k -> giá trị $S_{[i,n]}$ là prefix_max của dãy các tổng $S_{[j,n]}$

GIẢI THUẬT SONG SONG

- B1: Thực hiện tính tổng con trước Prefix Sum đối với cả mảng A, ta gọi là PSUM.
- B2: Thực hiện tính tổng con sau Suffix Sum đối với cả mảng A, ta gọi là SSUM.
- B3: Thực hiện tính dãy cực đại sau Suffix_max đối với mảng PSUM ta gọi là SMAX
- B4: Thực hiện tính dãy cực đại trước Prefix_max đối với mảng SSUM, ta gọi là PMAX
- B5: For 0<= k <= n-1 do in parallel
  - MS[k] = PMAX[k] – SSUM[k] + A[k]
  - MP[k] = SMAX[k] – PSUM[k] + A[k];
  - MSP[k] = MS[k] + MP[k] – A[k];
- B6: Tìm giá trị lớn nhất trong mảng MSP
- B7: Đưa ra kết quả cuối cùng

GIẢI THUẬT SONG SONG

Q	3	2	-7	11	10	-6	4	9	-6	1	-2	-3	4	-3	0	2
$PSUM$	3	5	-2	9	19	13	17	26	20	21	19	16	20	17	17	19
$SSUM$	19	16	14	21	10	0	6	2	-7	-1	-2	0	3	-1	2	2
$SMAX$	26	26	26	26	26	26	26	26	21	21	20	20	20	19	19	19
$PMAX$	19	19	19	21	21	21	21	21	21	21	21	21	21	21	21	21
M	26	26	26	28	28	28	28	28	23	23	22	22	22	21	21	21

- Maximum Subsequence Sum = Maximum(M) = 28 = Sum($Q_{4,8}$)

VẤN ĐỀ NẢY SINH

- i,j = ?
- Cách xác định:
  - Xác định max
  - Đặt 1 mảng B[i] = {0,1} là kết quả so sánh:
    - Nếu Max = M[i] -> B[i] = 1;
    - Nếu Max <> M[i] -> B[i] = 0;

VẤN ĐỀ NẢY SINH

- Xác định phần tử đầu tiên = 1

```
input    : B[1..n] = {0,1} ; n = 2^k.
output   : chỉ số i đầu tiên | B[i] = 1.
begin
      for i = 1 to n do in parallel
            if B[i] = 1 then VT[i] = i;          gán VT[i]
            else VT[i] = n+1;
      end parallel.
      for i = 1 to k do
            for j = 1 to n/2^i do in parallel
                  VT[j] = min {VT[2j-1], VT[2j]}   find min VT[i]
            end parallel
      end for
      return VT[1];
end.
```

VẤN ĐỀ NẢY SINH

B[1..8]	0	1	1	1	0	1	1	0
PB[1..8]	0	1	2	3	3	4	5	5
P[1..8]	0	1	2	3	0	0	0	0

VẤN ĐỀ NẢY SINH

```
input    : B[1..n] = {0,1}; n = 2^k ; i
output   : j max | B[i..j] = {1}
begin
        Tính PB[1..n] = Prefix_sum(B[1..n];

        for t = 1 to n do in parallel
                if PB[t] - PB[i] <> t - i then P[t] = 0;
                else P[t] = PB[t];
        end parallel

        for t = 1 to k do
                for v = 1 to n/2^t do in parallel
                        P[v] = max {P[2v-1], P[2v]}
                end parallel
        end for.

        for t = 1 to n do in parallel
                if PB[t] = P[1] then j = t;
        end parallel

end.
```

HẾT BÀI