**Example of usage of Prefix Sum**

**Compacting an Array**

![Example of usage of Prefix Sum](image)

- Given an array $A$ with many zeroes, compact it
  - Output an array $B$ such that all the values in $A$ not zero are at the beginning of $B$

*Any idea on the solution (first in sequential)?*

---

**Example of usage of Prefix Sum**

**Compacting an Array**

![Example of usage of Prefix Sum](image)

- Given an array $A$ with many zeroes, compact it
  - Output an array $B$ such that all the values in $A$ not zero are at the beginning of $B$

  - Initialize $B$ with zeroes
  - If $A[i] 
eq 0$ then $B[S[i]] = A[i]$

**How would you do it in parallel?**

---

**Example of usage of Prefix Sum**

**Compacting an Array**

![Example of usage of Prefix Sum](image)

- Given an array $A$ with many zeroes, compact it
  - Output an array $B$ such that all the values in $A$ not zero are at the beginning of $B$

  - Initialize $B$ with zeroes
  - If $A[i] 
eq 0$ then $B[S[i]] = A[i]$

**How can we use $S$?**

---

**Example of usage of Prefix Sum**

**Compacting an Array**

![Example of usage of Prefix Sum](image)

- Given an array $A$ with many zeroes, compact it
  - Output an array $B$ such that all the values in $A$ not zero are at the beginning of $B$

  - Hint: use an extra (binary) array $S$ such that
    - $S[i] = 0$ if $A[i] = 0$
    - $S[i] = 1$ if $A[i] 
eq 0$

**How can we use $S$?**

---

**Example of usage of Prefix Sum**

**Compacting an Array**

![Example of usage of Prefix Sum](image)

- Given an array $A$ with many zeroes, compact it
  - Output an array $B$ such that all the values in $A$ not zero are at the beginning of $B$

  - Hint: use an extra (binary) array $S$ such that
    - $S[i] = 0$ if $A[i] = 0$
    - $S[i] = 1$ if $A[i] 
eq 0$

**How can we use $S$?**

---

**Example of usage of Prefix Sum**

**Compacting an Array**

![Example of usage of Prefix Sum](image)

- Given an array $A$ with many zeroes, compact it
  - Output an array $B$ such that all the values in $A$ not zero are at the beginning of $B$

  - Hint: use an extra (binary) array $S$ such that
    - $S[i] = 0$ if $A[i] = 0$
    - $S[i] = 1$ if $A[i] 
eq 0$

**Algorithm COMPACT**

1. assign value 1 to $e$, and value 0 to the others
2. compute the PREFIX SUMS of these values, and store the results on $S$
3. begin
   begin
     $B(i):=0$; if $A(i)$ $\neq 0$ then $B( S(i) ):=A(i)$
   end
end
Algorithm COMPACT
1. assign value 1 to e and value 0 to the others
2. compute the PREFIX SUMS of these values, and store the results on S
3. begin constant time!
   for 1 ≤ i ≤ n pardo begin
     begin
     B(i): = 0; if A(i) ≠ 0 then B( S(i) ):=A(i)
     end
     end

Prefix Sums
- X = [2, 1, 4, 6, 9, 2, 1, 5]
- Let us reduce the problem to a n/2 array Y, where each element of Y is the sum in pairs of the elements in X
  - Y = [3, 10, 11, 6]
- Hint: Let us assume to have the prefix sum of Y
  - Z = [3, 13, 24, 30]

Can we compute the prefix sums of X from Z?

Prefix Sums
- X = [2, 1, 4, 6, 9, 2, 1, 5]
- Let us reduce the problem to a n/2 array Y, where each element of Y is the sum in pairs of the elements in X
  - Y = [3, 10, 11, 6]
- Hint: Let us assume to have the prefix sum of Y
  - Z = [3, 13, 24, 30]
- PrefixX = [2, Z[1], ....]

Prefix Sums
- X = [2, 1, 4, 6, 9, 2, 1, 5]
- Let us reduce the problem to a n/2 array Y, where each element of Y is the sum in pairs of the elements in X
  - Y = [3, 10, 11, 6]
- Hint: Let us assume to have the prefix sum of Y
  - Z = [3, 13, 24, 30]
- PrefixX = [2, Z[1], ....]

Prefix Sums
- X = [2, 1, 4, 6, 9, 2, 1, 5]
- Let us reduce the problem to a n/2 array Y, where each element of Y is the sum in pairs of the elements in X
  - Y = [3, 10, 11, 6]
- Hint: Let us assume to have the prefix sum of Y
  - Z = [3, 13, 24, 30]
- PrefixX = [2, Z[1], ....]
Prefix Sums

X = [2, 1, 4, 6, 9, 2, 1, 5]

Let us reduce the problem to a n/2 array Y, where each element of Y is the sum in pairs of the elements in X
- Y = [3, 10, 11, 6]

Hint: Let us assume to have the prefix sum of Y
- Z = [3, 13, 24, 30]

PrefixX =
[2, Z[1], Z[1]+X[3], ?]

Prefix Sums

X = [2, 1, 4, 6, 9, 2, 1, 5]

Let us reduce the problem to a n/2 array Y, where each element of Y is the sum in pairs of the elements in X
- Y = [3, 10, 11, 6]

Hint: Let us assume to have the prefix sum of Y
- Z = [3, 13, 24, 30]

PrefixX =

Prefix Sums

X = [2, 1, 4, 6, 9, 2, 1, 5]

Let us reduce the problem to a n/2 array Y, where each element of Y is the sum in pairs of the elements in X
- Y = [3, 10, 11, 6]

Hint: Let us assume to have the prefix sum of Y
- Z = [3, 13, 24, 30]

PrefixX =

Prefix Sums

X = [2, 1, 4, 6, 9, 2, 1, 5]

Let us reduce the problem to a n/2 array Y, where each element of Y is the sum in pairs of the elements in X
- Y = [3, 10, 11, 6]

Hint: Let us assume to have the prefix sum of Y
- Z = [3, 13, 24, 30]

PrefixX =

Prefix Sums (parallel version)

1. if n = 1 then s(1) := x(1) return
2. for 1 ≤ i ≤ n/2 pardo y(i) := x(2i - 1) * x(2i)
3. recursively compute the prefix sums of y(1), ..., y(n/2) and store them in z(1), ..., z(n/2)
4. for 1 ≤ i ≤ n pardo
   i. if i even then s(i) := z(i/2)
   ii. if i = 1 then s(1) := x(1)
   iii. if i odd then s(i) := z(i-1/2)*x(i)
end
**Prefix Sums via Doubling**

- Another interesting technique that can be used to solve the Prefix Sums is the Doubling
  - Iterative
  - A processing technique in which accesses or actions are governed by increasing powers of 2
  - That is, processing proceeds by 1, 2, 4, 8, 16, etc., doubling on each iteration

**Prefix Sums**

- At the first step, each X[i] is added to X[i+1]
  - X = [4, 9, 5, 2, 10, 6, 12, 8]
  - X1 = [4, 13, 14, 7, 12, 16, 18, 20]

How would you continue?

- At the second step, each X[i] is added to X[i+2]
  - X2 = [4, 13, 18, 20, 26, 23, 30, 36]

Next step?
Prefix Sums

- At the first step, each $X[i]$ is added to $X[i+1]$
  - At any time if an index exceeds $n$, the operation is suppressed
  - $X = [4, 9, 5, 2, 10, 6, 12, 8]$
  - $X_1 = [4, 13, 14, 7, 12, 16, 18, 20]$
- At the second step, each $X[i]$ is added to $X[i+2]$
  - $X_2 = [4, 13, 18, 20, 26, 23, 30, 36]$
- Doubling Time:
  - At step $k$, $X[i]$ is added to $X[i+2^{k-1}]$

Prefix Sums by Doubling

* Operation suppressed  # contains final sum

<table>
<thead>
<tr>
<th>0#</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7*</th>
</tr>
</thead>
<tbody>
<tr>
<td>0#</td>
<td>0.1#</td>
<td>1.2</td>
<td>2.3</td>
<td>3.4</td>
<td>4.5</td>
<td>5.6*</td>
<td>6.7*</td>
</tr>
<tr>
<td>0#</td>
<td>0.1#</td>
<td>0.1,2 #</td>
<td>0.1,2, 3#</td>
<td>1.2,3, 4</td>
<td>2.3,4, 5</td>
<td>3.4,5, 6</td>
<td>4.5,6, 7</td>
</tr>
<tr>
<td>0#</td>
<td>0.1#</td>
<td>0.1,2 #</td>
<td>0.1,2, 3#</td>
<td>0.1,2, 3,4#</td>
<td>0.1,2, 3,4,5 #</td>
<td>0.1,2, 3,4,5, 6#</td>
<td>0.1,2, 3,4,5, 6,7#</td>
</tr>
</tbody>
</table>

$p = n-1$  $T_p = O(\log n)$

Prefix Sums by Doubling

* Operation suppressed

<table>
<thead>
<tr>
<th>4</th>
<th>9</th>
<th>5</th>
<th>2</th>
<th>10</th>
<th>6</th>
<th>12</th>
<th>8*</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>13</td>
<td>14</td>
<td>7</td>
<td>12</td>
<td>16</td>
<td>18*</td>
<td>20*</td>
</tr>
<tr>
<td>4</td>
<td>13</td>
<td>18</td>
<td>20</td>
<td>26*</td>
<td>23*</td>
<td>30*</td>
<td>36*</td>
</tr>
<tr>
<td>4</td>
<td>13</td>
<td>18</td>
<td>20</td>
<td>30</td>
<td>36</td>
<td>48</td>
<td>56</td>
</tr>
</tbody>
</table>

How many steps do we need to finish?

Prefix Sums by Doubling

* Operation supressed  # contains final sum

<table>
<thead>
<tr>
<th>0#</th>
<th>1#</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7*</th>
</tr>
</thead>
<tbody>
<tr>
<td>0#</td>
<td>0,1#</td>
<td>1.2</td>
<td>2.3</td>
<td>3.4</td>
<td>4.5</td>
<td>5.6*</td>
<td>6.7*</td>
</tr>
<tr>
<td>0#</td>
<td>0,1#</td>
<td>0.1,2 #</td>
<td>0.1,2, 3#</td>
<td>1.2,3, 4</td>
<td>2.3,4, 5</td>
<td>3.4,5, 6</td>
<td>4.5,6, 7</td>
</tr>
<tr>
<td>0#</td>
<td>0,1#</td>
<td>0.1,2 #</td>
<td>0.1,2, 3#</td>
<td>0.1,2, 3,4#</td>
<td>0.1,2, 3,4,5 #</td>
<td>0.1,2, 3,4,5, 6#</td>
<td>0.1,2, 3,4,5, 6,7#</td>
</tr>
</tbody>
</table>

$p = ???$  $T_p = O(???)$
Prefix Sums by Doubling
• What about the Work (total number of operations)?
  – At the first step: $n-1$ operations
  – At the second step: $n-2$ operations
  – At the third step: $n-4$ operations
  – At the $k^{th}$ step: $n-2^{k-1}$ operations
• Total: $\sum_{k=0}^{\log n-1} (n-2^k)$ operations

Prefix Sums by Doubling
• What about the Work (total number of operations)?
  – At the first step: $n-1$ operations
  – At the second step: $n-2$ operations
  – At the third step: $n-4$ operations
  – At the $k^{th}$ step: $n-2^{k-1}$ operations
• Total: $\sum_{k=0}^{\log n-1} (n-2^k) = (n-1) + (n-2) + ... + (n-2^{\log n-1}) = (n \log n) - (1 + 2 + 4 + 2^{\log n-1})$
Prefix Sums by Doubling

- What about the Work (total number of operations)?
  - At the first step: n-1 operations
  - At the second step: n-2 operations
  - At the third step: n-4 operations
  - At the kth step: n-2k-1 operations

- Total: \( \sum_{k=0}^{\log n-1} (n-2^k) = (n-1) + (n-2) + \ldots + (n-2^{\log n-1}) = (n \log n) - (1 + 2 + 4 + 2^{\log n-1}) = (n \log n) - (2^{\log n} - 1) \)

List Ranking

INPUT

```
C  | a1 | a2 | a3 | a4 | a5 | a6 | a7 | a8 | a9 | a10
S  | 1   | 2   | 3   | n   | 7   |
```

CONTENT

SUCCESOR

OUTPUT

Array R such that R(i) is equal to the distance (rank) of item C(i) from the end of the list.

Prefix Sums

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Work</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sequential</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>Recursive</td>
<td>N</td>
<td>Log N</td>
</tr>
<tr>
<td>Doubling</td>
<td>N log N – (2^{\log N} - 1)</td>
<td>Log N</td>
</tr>
</tbody>
</table>

List Ranking

- Given a linked list, stored in an array, compute the distance of each element from the end (either end) of the list
  - Problem is similar to prefix sums, using all 1’s to sum
- Called Pointer Jumping (not doubling) when using pointers
- Don’t destroy original list!

Idea

```
a1, a2, a3, a4, a5, a6, a7, a8
R: 1 1 1 1 1 1 1 NIL
```

- At the beginning we initialize an array R (rank), that will contain the rank of each element
  - That is, the distance of each element from the end of the list

Idea

```
a1, a2, a3, a4, a5, a6, a7, a8
R: 1 1 1 1 1 1 1 NIL
```

- At the beginning, there are two elements with the correct rank... which ones??