Lecture 15 — October 11, 2017

Prof. Nodari Sitchinava *Scribe: Jipeng Huang, Christopher Yeager*

1 Homework 2 Solutions

1.1 Problem 1

---

for $i = 1$ *to* p **in parallel do**
  for $j = (i-1)\frac{n}{p} + 2$ *to* $i\frac{n}{p}$ **do**
    $A[i] = A[i-1] + A[i]$
  end for
end for
for $i = 1$ *to* p **in parallel do**
  $B[i] = A[i * \frac{n}{p}]$
  $prefix - sum(B)$
end for
for $i = 2$ *to* p **in parallel do**
  for $j = (i-1)\frac{n}{p} + 1$ *to* $i\frac{n}{p}$ **do**
    $A[j] = B[i-1] + A[j]$
  end for
end for

1.2 Problem 2

As we know, the total coins are $(C_1 + C_2 + ... + C_n)$ and there are m robbers which each robber share equally. Therefore, each robber should have total coins divided by m robbers. We allocate an array $A[n]$ to calculate the prefix-sum of coins so the $share = \lfloor \frac{A[n]}{m}$ and ith robber will take $(i-1) * share + 1$ to $i * share$ coins.

Algorithm 1 Split-loot(C, n , m)

$A[0...n]$
$C[1...n]$
for $i = 1$ *to* n **in parallel do**
  $A[i] = C[i]$
end for
$prefix - sum(A)$
$share = \lfloor \frac{A[n]}{m}$
$A[0] = 0$
for $j = 1$ *to* m **in parallel do**
  $output[j].l = predecessor((j-1) * share + 1, A)$
  $output[j].r = predecessor(j * share + 1, A)$
end for
if $A[output[j].l! = (j-1) * share + 1]$ **then**
  $output[j].l++$
end if
if $A[output[j].r! = (j-1) * share]$ **then**
  $output[j].r++$
end if
for $j = 1$ *to* m **in parallel do**
  $output[j].t_l = minshare, A[output[j].l] - (j-1) * share$
  if $output[j].t_l == share$ **then**
    $output[j].r++$
    $output[j].t_r = 0$
  else
    $output[j].t_r = j * share - A[output[j].r - 1]$
  end if
end for

We analyzed this algorithm that first for loop takes constant time and $O(n)$ work. The $prefix - sum(A)$ we talked in previous class and it takes $O(\log n)$ time and $O(n)$ work. Each *predecessor* function takes $O(1)$ time and $O(\log n)$ work. Therefore, the total time is $O(\log n)$ and work is $O(n)$ work.

1.3 Problem 3

We assume $a < b < c$ then we got: $min(min(a, b), c) = min(a, c) = a$ and $min(mina,(b, c)) = min(a, b) = a$.
Assume $a < c < b$ and we got: $min(min(a, b), c) = min(a, c) = a$ and $min(mina,(b, c)) = min(a, c) = a$.
Assume $b < a < c$ and we got: $min(min(a, b), c) = min(b, c) = b$ and $min(mina,(b, c)) = min(a, b) = b$.
Assume $b < c < a$ and we got: $min(min(a, b), c) = min(b, c) = b$ and $min(mina,(b, c)) = min(a, b) = b$.
Assume $c < a < b$ and we got: $min(min(a, b), c) = min(a, c) = c$ and $min(mina,(b, c)) = min(a, c) = c$.

Assume $c < b < a$ and we got: $min(min(a,b),c) = min(b,c) = c$ and $min(mina,(b,c)) = min(a,c) = c$.
Therefore, the min is associative operator. since $min(I_{min}, x) = x$, so its identity is $I_{min} = \infty$

Algorithm 2 Minima(A[0...n-1])

```
if n == 0 then
  return A[0]
else
  for i = 0 to n/2 - 1 in parallel do
    B[i] = min(A[2i], A[2i + 1])
  end for
  Minima(B[0... n/2 - 1])
end if
```
