Activity : Extracting Parallelism (1D cases)

Extracting dependency from code is an almost automatic process. You need to choose a granularity. But once that is chosen, the entire analysis follows.

In the whole activity, you should express the metrics in complexity notation as a function of the parameters of the functions.

1 Transform

Consider the transform function:

```
void transform (int* a, int* b, int n) {
  for (int i=0; i<n; ++i)
    b[i] = f(a[i]);
}
```

Task Bi

Question: What is the complexity of this function?

Question: Extract the dependencies. Assume the call to `f` cost O(1). Assume calls to `f` are always independent. (Note: Yes this problem is VERY simple!)

O(n)

Question: What is the width?

Question: What is the work?

n

Question: What is the critical path? What is its length?

n

any 'B' node can be the critical path. 1

2 Reduce

Consider the reduce function:

```
template<typename T, typename op>
T reduce (T* array, size_t n) {
  T result = array[0];                          Task A
  for (int i=1; i<n; ++i)
    result = op (result, array[i]);             Task Bi
  return result;
}
```

Do not be scared by the syntax! In C++, templates allow you to replace types and values in a piece of code by a type or a value known at compilation time. This is similar to generics in Java.

So if you define `T` as `int` and `op` as `sum`, it boils down to computing the sum of the array. You could use `op` as `max` and compute the maximum value of the array.

Assume `T` is `int` and `op` is `sum`.

Question: What is the complexity of this function? O(n)

Question: Extract the dependencies. 1

Question: What is the width? n

Question: What is the work? a linear chain of all tasks is the critical path. n

Question: What is the critical path? What is its length?

3 Prefix sum

Prefixsum is an algorithm that has many uses in parallel computing. The algorithm computes $pr[i] = \sum_{j<i} arr[j], \forall 0 \leq i \leq n$ and is often written sequentially:

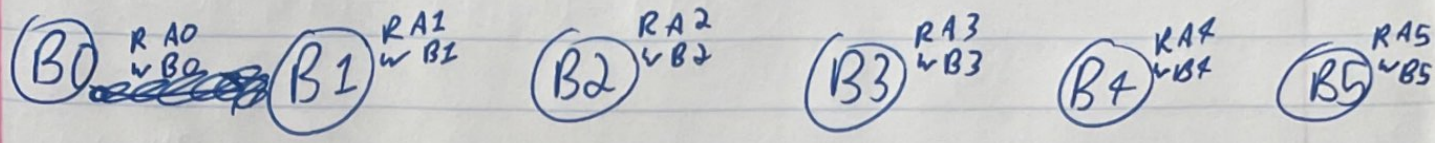
```
void prefixsum (int* arr, int n, int* pr) {
  pr[0] = 0;
  for (int i=0; i<n; ++i)
    pr[i+1] = pr[i] + arr[i];
}
```

Question: What is the complexity of this function? O(n)
Question: Extract the dependencies.
Question: What is the width? n
Question: What is the work? n
Question: What is the critical path? What is its length? a linear chain of all tasks is the critical path. n

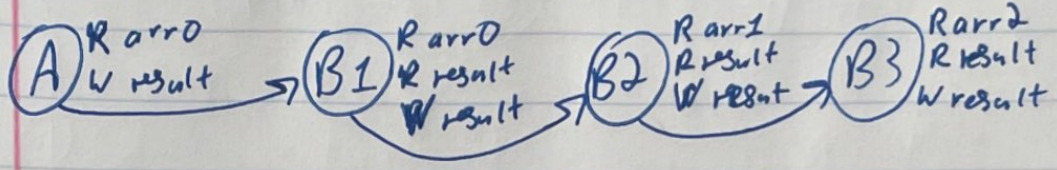
extraction 1

1 Transform

n=5



2 Reduce



3 Prefix Sum