

---

# COMP 322: Fundamentals of Parallel Programming

## Lecture 6: Memoization

Instructors: Vivek Sarkar, Mack Joyner  
Department of Computer Science, Rice University  
{vsarkar, mjoyner}@rice.edu

<http://comp322.rice.edu>

---

COMP 322

Lecture 6

23 January 2017



---

### Worksheet #5: Computation Graphs for Async-Finish and Future Constructs

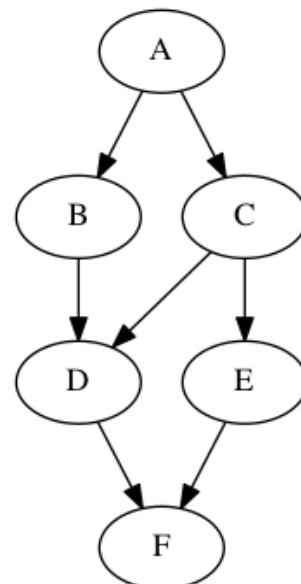
---

1) Can you write pseudocode with async-finish constructs that generates a Computation Graph with the same ordering constraints as the graph on the right? If so, provide a sketch of the program.

No. Finish cannot be used to ensure that D waits for both B and C, while E waits only for C.

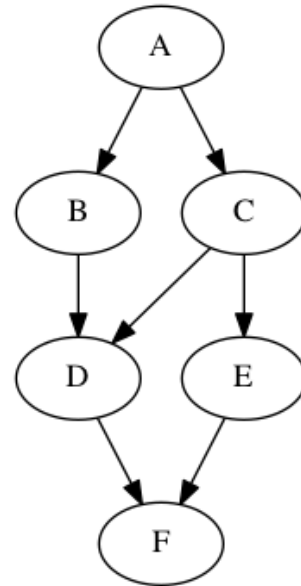
2) Can you write pseudocode with future async-get constructs that generates a Computation Graph with the same ordering constraints as the graph on the right? If so, provide a sketch of the program.

Yes, see program sketch with void futures. A dummy return value can also be used.



## Worksheet #5 solution (contd)

```
1. HjFuture<String> A = future(() -> {
2.     return "A"; });
3. HjFuture<String> B = future(() -> {
4.     A.get(); return "B"; });
5. HjFuture<String> C = future(() -> {
6.     A.get(); return "C"; });
7. HjFuture<String> D = future(() -> {
8.     // Order of B.get() & C.get() doesn't matter
9.     B.get(); C.get(); return "D"; });
10. HjFuture<String> E = future(() -> {
11.     C.get(); return "E"; });
12. HjFuture<String> F = future(() -> {
13.     D.get(); E.get(); return "F"; });
14. F.get();
```



## Background: Functional Programming

- Eliminate side-effects
  - emphasizes functions whose results that depend only on their inputs and not on any other program state
  - calling a function,  $f(x)$ , twice with the same value for the argument  $x$  will produce the same result both times

**Helpful Link:** [http://en.wikipedia.org/wiki/Functional\\_programming](http://en.wikipedia.org/wiki/Functional_programming)



## Example: Binomial Coefficient

---

- The coefficient of the  $x^k$  term in the polynomial expansion of the binomial power  $(1 + x)^n$
- Number of sets of  $k$  items that can be chosen from  $n$  items
- Indexed by  $n$  and  $k$ 
  - written as  $C(n, k)$
  - read as “ $n$  choose  $k$ ”
- Factorial Formula:  $C(n, k) = \left( \frac{n!}{k!(n-k)!} \right)$
- Recursive Formula  
 $C(n, k) = C(n - 1, k - 1) + C(n - 1, k)$   
Base cases:  $C(n, n) = C(n, 0) = C(0, k) = 1$

**Helpful Link:** [http://en.wikipedia.org/wiki/Binomial\\_coefficient](http://en.wikipedia.org/wiki/Binomial_coefficient)



## Example: Binomial Coefficient (Recursive Sequential version)

---

```
1. int choose(int N, int K) {
2.     if (N == 0 || K == 0 || N == K) {
3.         return 1;
4.     }
5.     int left = choose (N-1, K - 1);
6.     int right = choose (N-1, K);
7.     return left + right;
8. }
```



## Example: Binomial Coefficient (Parallel Recursive Pseudocode)

```
1. Integer choose(int N, int K) {
2.     if (N == 0 || K == 0 || N == K) {
3.         return 1;
4.     }
5.     future<Integer> left =
6.         future { return choose (N-1, K-1); }
7.     future<Integer> right =
8.         future { return choose (N-1, K); }
7.     return left.get() + right.get();
8. }
```

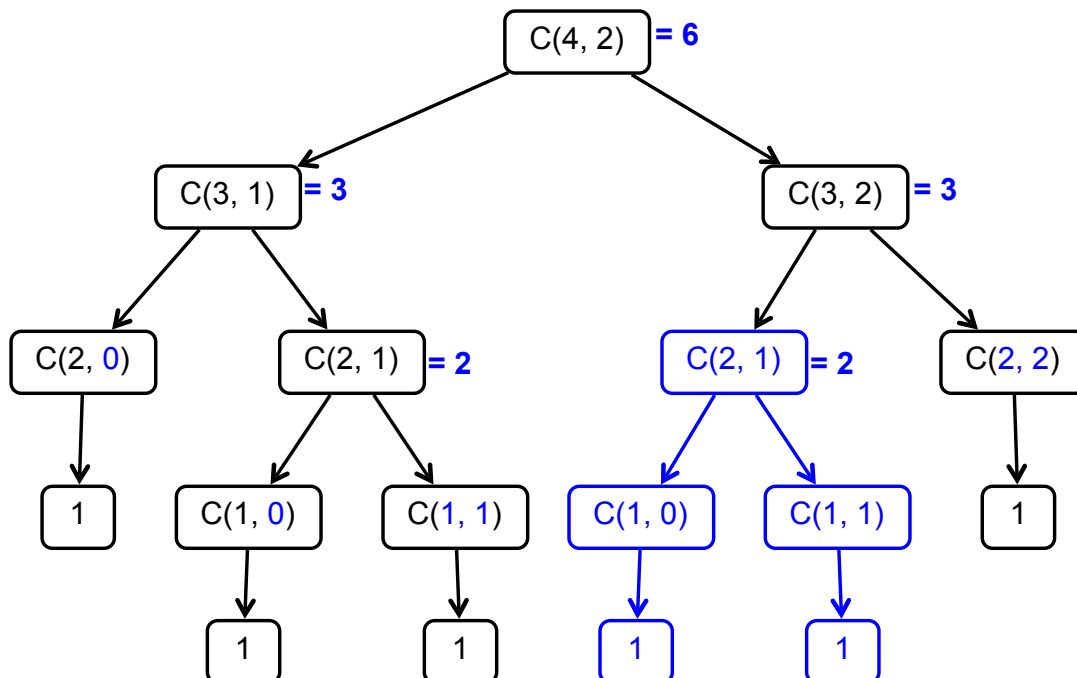
- Use of futures supports incremental parallelization with low developer effort

7

COMP 322, Spring 2017 (V. Sarkar, M. Joyner)



## What inefficiencies do you see in the recursive Binomial Coefficient algorithm?



8

COMP 322, Spring 2017 (V. Sarkar, M. Joyner)



# Memoization

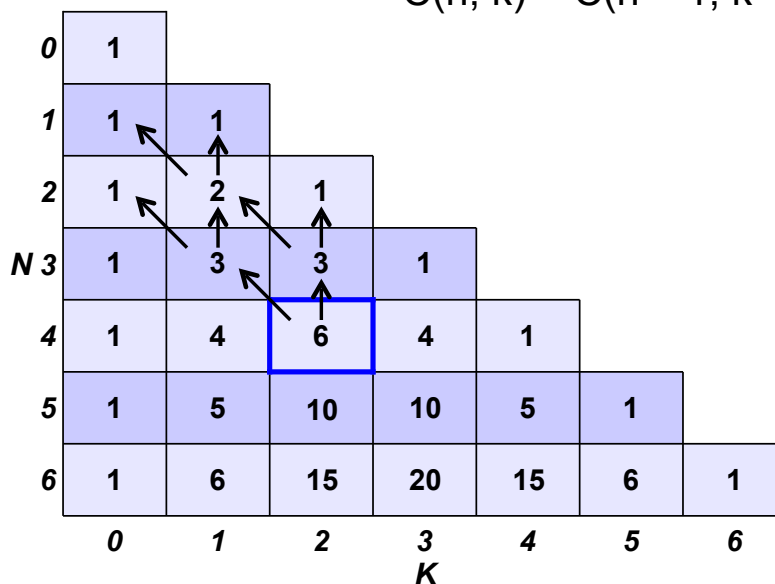
- Memoization - saving and reusing previously computed values of a function rather than recomputing them
  - A optimization technique with space-time tradeoff
- A function can only be memoized if it is *referentially transparent*, i.e. functional
- Related to caching
  - memoized function "remembers" the results corresponding to some set of specific inputs
  - memoized function populates its cache of results transparently on the fly, as needed, rather than in advance

Helpful Link: <http://en.wikipedia.org/wiki/Memoization>



## Pascal's Triangle is an example of Memoization

$$C(n, k) = C(n - 1, k - 1) + C(n - 1, k)$$



## Example: Binomial Coefficient (sequential memoized version)

---

```
1. final Map<Pair<Int, Int>, Int> cache = new ...;

2. int choose(int N, int K) {
3.     Pair<Int, Int> key = Pair.factory(N, K);
4.     if (cache.contains(key)) {
5.         return cache.get(key);
6.     }
7.     if (N == 0 || K == 0 || N == K) {
8.         return 1;
9.     }
10.    int left = choose (N - 1, K - 1);
11.    int right = choose (N - 1, K);
12.    int result = left + right;
13.    cache.put(key, result);
14.    return result;
15. }
```

---

11

COMP 322, Spring 2017 (V. Sarkar, M. Joyner)



## Example: Binomial Coefficient (parallel memoized version w/ futures)

---

```
1. final Map<Pair<Int, Int>, future<Integer>> cache = new ...;
2. Integer choose(final int N, final int K) {
3.     final Pair<Int, Int> key = Pair.factory(N, K);
4.     if (cache.contains(key)) {
5.         return cache.get(key).get();
6.     }
7.     future<Integer> f = future {
7.         if (N == 0 || K == 0 || N == K) return 1;
8.         future<Integer> left = future { return choose (N-1, K-1); }
9.         future<Integer> right = future { return choose (N-1, K); }
12.        return left.get() + right.get();
13.    }
14.    cache.put(key, f);
15.    return f.get();
16. }
```

- Assumes availability of a “thread-safe” cache library, e.g., ConcurrentHashMap

---

12

COMP 322, Spring 2017 (V. Sarkar, M. Joyner)



# Announcements & Reminders

---

- **IMPORTANT:**
  - [Watch video & read handout for topic 2.3 for next lecture on Wednesday, Jan 25th](#)
- HW1 was posted on the course web site (<http://comp322.rice.edu>) on Jan 11th, and is due on Jan 25th
- Quiz for Unit 1 (topics 1.1 - 1.5) is due by Jan 27th on Canvas
- See course web site for all work assignments and due dates
- Use Piazza (public or private posts, as appropriate) for all communications re. COMP 322
- See [Office Hours](#) link on course web site for latest office hours schedule. Group office hours are now scheduled during 3pm - 4pm on MWF in DH 3092 (default room but alternate room may need to be used on some days — an announcement will be made in the lecture on those days)

