Computer Science

An Introduction and Some Advanced Concepts Too!

Prof. Paul Sivilotti

Dept. of Computer & Info. Science
The Ohio State University

paolo@cis.ohio-state.edu

Computers Are All Around

The 1st Computer Scientist

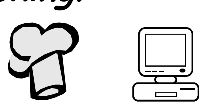
Ada Byron King,
Countess of Lovelace
1815-1852

Computers and Programs

- Computer: a device that "computes"
  - Takes inputs, produces output
  - Becoming smaller, faster, cheaper
- Program: sequence of instructions
  - How to produce the output
  - Must be specific
  - Becoming larger and more complicated!

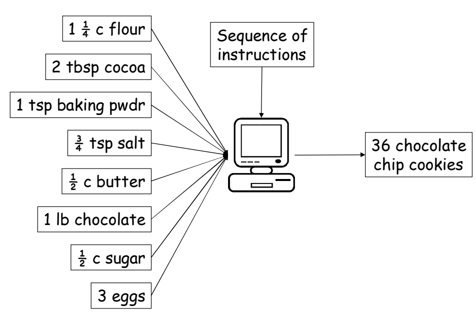
Now We're Cooking!

- Computer = chef
- Program = recipe



1. Preheat oven to 350°
2. Sift together flour, cocoa, baking powder, salt
3. Melt 1/2c butter and 1lb chocolate
4. Stir 1/2c sugar into chocolate mixture
5. Sir in 3 large eggs
6. Stir in dry ingredients
7. Add chocolate chunks
8. Form into rounded balls (1T each)
9. Bake 10 min

Computing Choc. Cookies

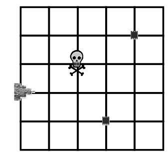


Software Engineering

- A software engineer builds *programs*
  - Design, develop, test, modify, maintain
- Program requirements?
  - Most important: ingredients and final dish!
  - (Also: time, space)
- For the same requirements, many solutions
  - Good recipes are *correct*
  - Good recipes are *easy to understand*
  - Good recipes are *easy to change*

Module I: Write a Program

- Robots on a grid
- Stack of index cards
  - Each is an instruction for robot
  - fwd 1, turn, pick up, …
- Requirements:
  - Robot initial position
  - Robot goal(s)
  - Other constraints
- Your task: write a program for the robot!
  - A sequence of cards
  - Robot follows program



Now We're Cooking!

- Computer = chef
- Program = recipe

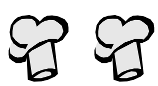

1. Preheat oven to 350°
2. Sift together flour, cocoa, baking powder, salt
3. Melt 1/2c butter and 1lb chocolate
4. Stir 1/2c sugar into chocolate mixture
5. Sir in 3 large eggs
6. Stir in dry ingredients
7. Add chocolate chunks
8. Form into rounded balls (1T each)
9. Bake 10 min

Time Required

1.	Preheat oven	
2.	Dry ingredients	5 min
3.	Melt chocolate	6 min
4.	Add sugar	2 min
5.	Add eggs	1 min
6.	Combine wet & dry	1 min
7.	Add chocolate chunks	
8.	Form into balls	10 min
9.	Bake	10 min

Total: 35 min

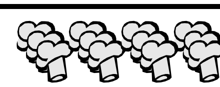
Two Chefs



1.	Melt chocolate/Dry ingredients	6 min	5 min
2.	Add sugar	2 min	
3.	Add eggs	1 min	
4.	Combine wet & dry	1 min	
5.	Add chocolate chunks		
6.	Form into balls	5 min	5 min
7.	Bake	10 min	

Total: 25 min
(> 35/2)

Lots of Chefs

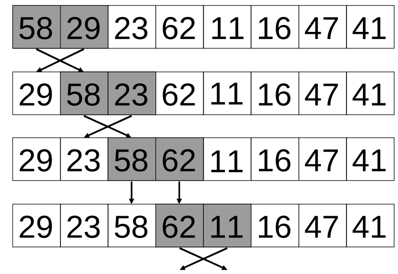


1.	Melt choc/Dry ingr	6 min	1 min	1 min	2 min	1 min
2.	Add sugar	2 min				
3.	Add eggs	1 min				
4.	Combine wet & dry	1 min				
5.	Add choc chunks					
6.	Form into balls	0 min	0 min	0 min	0 min	…
7.	Bake	10 min				

Total: 20 min

Diminishing Returns

A Sorting Program: BubbleSort



Time Required (Bubblesort)

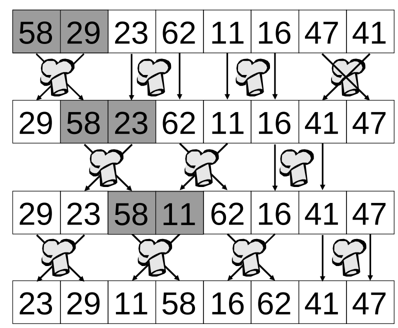
- After going down the whole list, what can we say about the result?

29	23	58	11	16	47	41	62

- After going down the whole list twice, what can we say about the result?

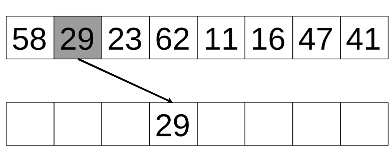
23	29	11	16	47	41	58	62

A Faster Program



Another Parallel Program

- To figure out where a number goes: *Count how many numbers are smaller!*



- Do this for each number in parallel!

Summary

- Sequential programs are *slow*
- Use as many chefs as possible!
  - Every day they get faster and cheaper
- New recipes are needed that scale efficiently
  - Note: increased complexity of recipe
  - So we need new ways to design these complex recipes

Module II: Parallel Program

- Sequential program:
  - Bubble Sort
- Parallel programs:
  - Even-Odd Transposition Sort
  - Radix Sort

Fault-Tolerance

- Sometimes, when a program runs, things go wrong: a *fault*
- Faults are rare, but do occur
  - Oven doesn't work!
    - Cookies end up gloopy
  - No eggs in pantry!
    - Chef stops, doesn't know what to do (no cookies)
  - Use salt by accident instead of sugar!
    - Cookies end up gross
- "Fault-tolerant": a program that still does the right thing, despite faults
  - The program heals itself, and recovers

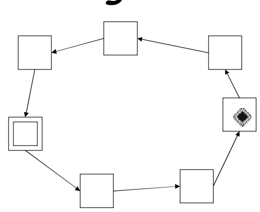
Making Sure the Cookies Turn Out

- How can you improve the odds of getting a good batch of cookies?
- Answer: use many chefs!
  - Each chef makes a complete recipe
  - Pick the best batch
- Even if some chefs experience faults, most will not
- "Triple Modular Redundancy"
  - Simple, expensive

Tokens for Taking Turns

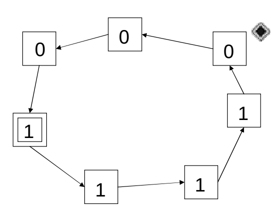
- Consider all the chefs across the city
- Say they need to take turns
  - Only one can be on vacation at a time
- How do they coordinate when to go on vacation?
- Solution: use a "token"
  - Pass token around
  - Rule: *If you have the token, you can go on vacation*

Tokens Ring



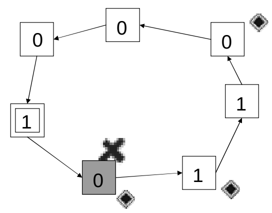
- Problem: What about faults?
  - What happens if token is lost?
  - One fault means disaster!

Prevent Loss of Token



- Rule (for most chefs):
  if left neighbor is different from me, then I have the token
  - Make my number equal to that neighbor's

Fault: Corruption of Values



- Problem: multiple tokens in ring
  - Tokens chase each other around ring
  - One fault means disaster

Dijkstra's Token Ring

- Use more than 2 values!
- Same rule
  - If left neighbor different from me:
    - I have the token! (use it)
    - Change my value to be equal to neighbor
- Again, one chef is special
  - If left neighbor *same* as me
    - I have the token (use it)
    - Change my value to be *one bigger*

Module III: Token Rings

- Form a ring
  - Each person has number cards
  - Each person has a chime
- When you get the token:
  - *Play your chime*
  - Then change your number
- We'll run different versions
  - I'll introduce "faults" and see if you can recover!

Take-Home Messages

- Computer program: a sequence of instructions
  - A recipe for a chef
- Software engineering: how to design programs
  - Recipe requirements: ingredients and final dish
  - Recipe design: correct, easy to understand and modify
- Parallel programming: lots of chefs in the kitchen
  - Sequential programs are slow, parallel programs are fast!
- Fault tolerance: programs can heal themselves
  - Redundancy
  - Distributed programs