C Programming: Sorting, structs, linked lists, pass-by-value vs. pass-by-reference

Yipeng Huang

Rutgers University

February 3, 2022

Table of contents

Quiz due tonight & programming assignments & next quiz

Quiz due tonight

- Due tonight, 11:59 pm.
- Two tries, 45 minutes each.
- Individual work. Open book. Experiment on iLab.

Programming assignments

- PA0 grades released.
- PA1 due Tuesday.
- Goal for Today: More stepping stones for beating PA1.

Next quiz

- Same quiz schedule as this week; spans Tuesday to Thursday
- Two tries, 30 or 45 minute time limit TBD.
- Individual work. Open book. Experiment on iLab.

Table of contents

`sortJobs_selection.c`: Basic sorting approach

Notice the nested loop:

- Outer nested loop iterating over timeslots.
- Inner nested loop iterating the jobs read from the input files.

So, input file is read multiple times, once for each timeslot.

Table of contents

`sortJobs_insertion.c`: Using structs to build a linked list to support insertion sort

File listing the jobs is read only once

- A linked list stores the jobs in sorted order.
- After linked list built, iterating and reading the linked list gives sorted order.

To build linked list:

- C `structs`
- Syntax for accessing struct elements.
- mallocing nodes for linked list.
- freeing linked list.

Table of contents

Why pointers?

Pointers underlie almost every programming language feature:

- arrays
- pass-by-reference
- data structures

Vital reason why C is a low-level, high-performance, systems-oriented programming language (why we use it for this class, computer architecture).

git pull

- From the folder 2022_0s_211, type: `git pull`.
- By now we have several example codes: `isPrime.c`, `numList.c`, `pointers.c`, `dotProduct.c`.
- This hands-on-lab is in `pointers.c`.

Lesson 7: Passing-by-value

Using stack and heap picture, understand how pass by value and pass by reference are different.

- C functions are entirely pass-by-value
- `swap_pass_by_values()` doesn't actually succeed in swapping two variables.

Lesson 8: Passing-by-reference

Using stack and heap picture, understand how pass by value and pass by reference are different.

- You can create the illusion of pass-by-reference by passing pointers
- `swap_pass_by_references()` does succeed in swapping two variables.

Lesson 9: Passing an array leads to passing-by-reference

Table of contents

`dynamicProgramming.c`: Minimum number of multiplies needed for matrix chain multiplication

Cost of multiplying matrices: the number of multiplies

- $A_{l \times m} \times B_{m \times n}$
- Generally speaking, $l \times m \times n$ number of multiplies
- (Well-kept secret: fewer multiplications possible, see Strassen's algorithm)

`dynamicProgramming.c`: Minimum number of multiplies needed for matrix chain multiplication

$$A \times B \times C = \begin{bmatrix} a_{0,0} & a_{0,1} \end{bmatrix}_{1\times 2} \times \begin{bmatrix} b_{0,0} \\ b_{1,0} \end{bmatrix}_{2\times 1} \times \begin{bmatrix} c_{0,0} & c_{0,1} \end{bmatrix}_{1\times 2}$$

Parenthesization 1: 4+4 = 8 multiplies

$$\begin{aligned} A \times (B \times C) &= \begin{bmatrix} a_{0,0} & a_{0,1} \end{bmatrix}_{1\times 2} \times \begin{bmatrix} b_{0,0}c_{0,0} & b_{0,0}c_{0,1} \\ b_{1,0}c_{0,0} & b_{1,0}c_{0,1} \end{bmatrix}_{2\times 2} \\ &= \begin{bmatrix} \left(a_{0,0}b_{0,0}c_{0,0} + a_{0,1}b_{1,0}c_{0,0}\right) & \left(a_{0,0}b_{0,0}c_{0,1} + a_{0,1}b_{1,0}c_{0,1}\right) \end{bmatrix}_{1\times 2} \end{aligned}$$

Parenthesization 2: 2+2 = 4 multiplies

$$\begin{aligned} (A \times B) \times C &= \left(a_{0,0}b_{0,0} + a_{0,1}b_{1,0}\right) \times \begin{bmatrix} c_{0,0} & c_{0,1} \end{bmatrix}_{1\times 2} \\ &= \begin{bmatrix} \left(a_{0,0}b_{0,0}c_{0,0} + a_{0,1}b_{1,0}c_{0,0}\right) & \left(a_{0,0}b_{0,0}c_{0,1} + a_{0,1}b_{1,0}c_{0,1}\right) \end{bmatrix}_{1\times 2} \end{aligned}$$

`dynamicProgramming.c`: Minimum number of multiplies needed for matrix chain multiplication

$A \times B \times C \times D$

First partitioning

- $A(BCD)$; but what is cost of finding (BCD)? Needs decomposition.
- $(AB)(CD)$
- $(ABC)D$; but what is cost of finding (ABC)? Needs decomposition.

Second partitioning

- $A(B(CD))$
- $A((BC)D)$
- $(AB)(CD)$
- $(A(BC))D$
- $((AB)C)D$

Table of contents

`matMul.c`: Function for matrix-matrix multiplication

What to pay attention to

- How `matMulProduct` result is given back to caller of function.
- How and where memory is allocated and freed.