16.070 Introduction to Computers and Programming

April 18 **Recitation 10** **Spring 2002**

Topics:

- Corrections to Rec 9
  - Merge Sort
  - Big O Notation
- Final Project
- Serial I/O
- Other

Corrections to Recitation 9

Merge Sort

a. Split the array into two parts
b. Merge Sort each part
c. Shuffle the two parts back into one array
d. Code: …
e. Recursive Algorithm (see side topic)
f. Faster than Bubble or Selection Sort. O(2NlogN)

For any two sorted lists of numbers, you merge them like so:

1. Look at the first element in each list. Whichever is less, move that smaller element to the head of a new list.
2. The second element in that list is now the first element, so continue until the new list is a sorted version of the previous lists.

Example: Original Array = 9 6 2 1 8 4 7 3

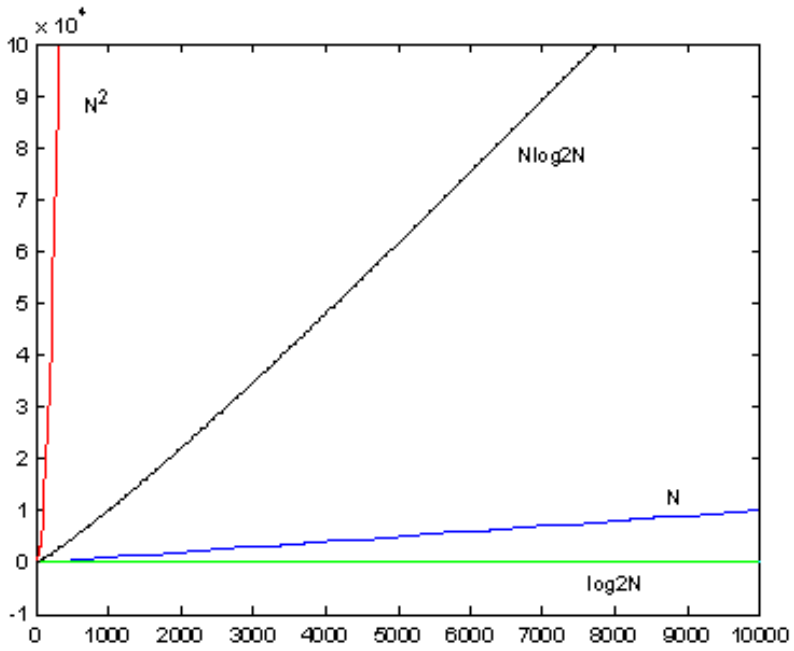
```
Split1:    {9 6 2 1}              {8 4 7 3}
Split2:  {9 6}   {2 1}         {8 4}     {7 3}
Split3: {9} {6} {2} {1}       {8} {4}   {7} {3}

Merge3: {9} {6} = {}       {2} {1} = {}       {8} {4} = {}       {7} {3} = {}
        {9} {  } = {6}     {2} {  } = {1}     {8} {  } = {4}     {7} {  } = {3}
        {  } {  } = {6 9}  {  } {  } = {1 2}  {  } {  } = {4 8}  {  } {  } = {3 7}
Merge2: {6 9} {1 2} = {}                      {4 8} {3 7} = {}
        {6 9} {   2} = {1}                    {4 8} {   7} = {3}
        {6 9} {     } = {1 2}                 {   8} {   7} = {3 4}
        {   9} {     } = {1 2 6}              {   8} {     } = {3 4 7}
        {     } {     } = {1 2 6 9}           {     } {     } = {3 4 7 8}
Merge1: {1 2 6 9} {3 4 7 8} = {}
        {   2 6 9} {3 4 7 8} = {1}
        {      6 9} {3 4 7 8} = {1 2}
        {      6 9} {   4 7 8} = {1 2 3}
        {      6 9} {      7 8} = {1 2 3 4}
        {        9} {      7 8} = {1 2 3 4 6}
        {        9} {        8} = {1 2 3 4 6 7}
        {        9} {          } = {1 2 3 4 6 7 8}
        {          } {          } = {1 2 3 4 6 7 8 9}
```

Order of Magnitude (Big O)

O = order of magnitude upper bound on execution time
How many operations does it take to do a task? As you add elements, how does the number of operations change?

Which Big O is faster?

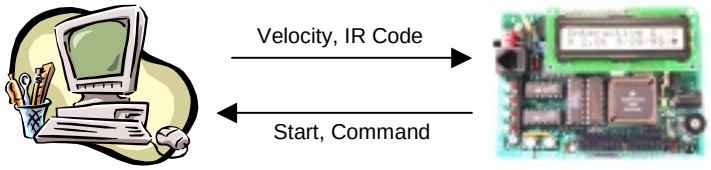
$O(N^2)$ > O(NlogN) > O(N) > O(logN) > O(1)



Final Project

Interfaces

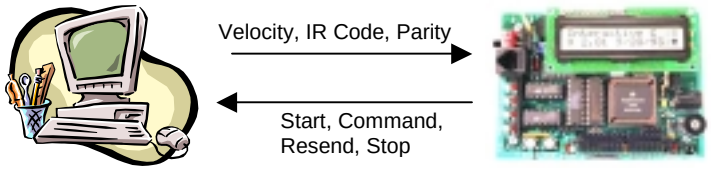
Part A



Workstation running Visual C:
Test Harness

Student Handy Board:
Controller Software

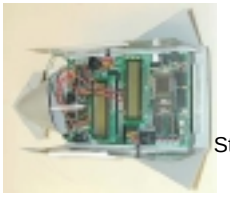
Part B



Workstation running Visual C:
Simulator

Student Handy Board:
Controller Software

Part C



Velocity, IR Code, Parity

Start, Command, Resend, Stop

Vehicle with On-Board Handy Board:
Provided Code

Student Handy Board:
Controller Software

Test Harness

What is it?

The test harness must

- Receive one byte at a time from the Handy Board
- Display/process received byte
  - Wait for the start byte
  - Display command
- Send one test byte at a time to the Handy Board (after the start byte is received)

Where does the test byte come from?

There are several test conditions listed in the write up. You should create three more. These test bytes can either be hard coded into the test harness, or the test harness can prompt the user for the encoded byte, or the test harness can prompt the user for velocity and IR information. If you use the last method, make sure your encoding scheme is correct.

How do you communicate with the Handy Board?

See next topic…

Synchronization Problems?

The Handy Board will wait until a byte is actually received.
Visual C will check the serial line and then move to the next line of code even if it received no data.

Serial Port Communication Example

Transmit a test bit from the workstation to the handy board

(see code at http://web.mit.edu/16.070/www/recitation/R10/ws.c and …/R10/hb.c)

Transmit from the workstation

1. Include serial library
2. Store test bit
3. Call serial communication function
4. Compile and Run

Receive on the handy board

1. Load Serial Libraries
2. Call disable pcode function
3. Call serial receive function
4. Process received bit
5. Call enable pcode function
6. Load main
7. Close IC

Workstation Code: Fill in the /**********/ 's

```
/*******************
* Recitation 10
* Workstation I/O
* Kay Sullivan 4/17
*********************/

#include <stdio.h>

/* Include library with serial communication functions */
/*******************************/

#define MAX_VEL 13.0f       /* What are the valid velocity values in the final project? */

/* function prototypes */
unsigned char write_telem(float vel, int ir);
int decode_ir(int byte);
float decode_vel(int byte);

/**********************
* MAIN - test harness
* for write_telem
**********************/
int main(void)
{
        float input_vel = 0.0f;
        int input_ir = 0;

        /* variable to hold byte to send */
        /*******************************/

        int stop = 0;
        while(!stop)
        {
                /* get inputs */
                input_vel = 100.0f;
                while((input_vel < -13.0f) || (input_vel > 13.0f))
                {
                        printf("Enter Velocity (-13.0 to 13.0): ");
                        scanf("%f", &input_vel);
                }/*end while*/

                input_ir = 0;
                while((input_ir != 1) && (input_ir != 2) && (input_ir != 3))
                {
                        printf("Enter IR Code (1=visible, 2=not visible, 3=docked): ");
                        scanf("%d", &input_ir);
                }/*end while*/

                /* make outbyte */
                /*******************************/

                /* send to HB */
                /*******************************/

                /* echo print to screen */
                printf("outbyte: %d  vel: %lf   ir: %d\n", out_byte, decode_vel(out_byte),
decode_ir(out_byte));

                printf("Stop? ");
                scanf("%d", &stop);
        }/*end while*/
        return 0;
}/*end main*/
```

Handy Board Code: Fill in the /**********/ 's

```
/*******************
* Recitation 10
* Handy Board I/O
* Kay Sullivan 4/17
*********************/

/*******************
* MAIN
*******************/
int main(void)
{
	/* variable to hold received byte */
	/**********************************/

	float hb_vel = 0.0;
	int hb_ir = 0;

	/* get control of the serial port */
	printf("Grabbing serial port\n");
	/**********************************/

	printf("Ready to receive\n");
	while(!stop_button())
	{
		/* listen on serial line */
		/*************************************/

		/* process any received data */
		if(inbyte != 0)
		{
			printf("Got data... processing...\n");
			hb_vel = decode_vel(inbyte);
			hb_ir = decode_ir(inbyte);
			printf("Vel: %f    IR: %d\n", hb_vel, hb_ir);
		}/*end if*/

	}/* end while */

	/* return control of the serial port */
	printf("Releasing serial port\n");
	/**********************************/

	return 0;
}/*end main*/
```