# Structuring Data & Building Algorithms
# Tutorials

白小明
**Jonathon David White**
元智大学光电系
**R70740, R70723**
**WhiteJD@XiaoTu.com**

**修改:中華民國 103 年 3 月 6 日 16 時 57 分 28 秒**
**Modified AD14 年 3 月 6 日 (Rev 156)**

**Location: Www.sdba.info/tut/**

# Table of Contents

| Date | | Class ID | | | TA Stamp |
|---|---|---|---|---|---|
| Time | | Student ID | | | |
| Name | | | | | |

# 0 Using the Editor to Write a Simple Program

## 0.1 Customizing the look and feel of jEdit

| Task | OK |
|---|---|
| Gutter: Line Numbers On | |
| Text Area Font: 18 Point Courier New | |
| Tab Width=4 Indent Width =4 | |
| Character Encoding: US-ASCII | |
| Backup: Save 99 backups,<br>Backup: Frequency = 60 sec<br>Backup: Directory: C:\backup | |
| Spit the screen to get 2 windows | |

```
/* sdba.c simple program */
#include <stdio.h>
int main(void){
  int j, jon, jane;
  jon=1;
  for(j=0;j<5;j++){
    fprintf(stdout,"Hi...");
    jon++;
    jane+=jane+5*jon;
  }
return(0);
}
```

## 0.2 Using the gcc compiler

1. ***Enter the program (sdba.c) into your computer using jEdit***
2. ***Compile, Link & Run the program. Make sure the program runs correctly.***

```
gcc -c -ansi -Wall sdba.c
gcc -o s.exe sdba.o
s
```

3. ***Change the program and record the resulting error messages below***

| | Change made | Error/Warning Message Received from Compiler |
|---|---|---|
| 1 | | |
| 2 | | |
| 3 | | |
| 4 | | |
| 5 | | |

## 0.3 Use the Gnu Debugger (gdb) to step through program

1. ***Compile the program with the `-g` option***
2. ***Start the `gdb` debugger.***
3. ***Step through the program, display the value of `j`, `jon` and `jane` at each step***

When you have completed the work, ask the TA to check your work. Be prepared to demonstrate compiling, linking and running the code in the debugger.

| Date | | Class ID | | | TA Stamp |
|---|---|---|---|---|---|
| Time | | Student ID | | | |
| Name | | | | | |

# 1 Variables & Pointers

## 1.1 Variables; Declaring, Initializing, Using

***1. Which of the following variable declarations are valid?***

| | OK? | | OK? | | OK? |
|---|---|---|---|---|---|
| `double jon;` | O | `float ABC;` | | `int dogs2;` | |
| `double %c;` | | `float int;` | | `long cat_hi;` | |
| `char Chong;` | | `int 2cat;` | | `float nMan;` | |

***2. Which of the following assignments are valid, dangerous or wrong? Why?***

Given these declarations: `int ann; char sue; double jim;`

| | OK? | | OK? | | OK? |
|---|---|---|---|---|---|
| `ann=1956` | X (no ;) | `ann=sue;` | | `ann=jim;` | |
| `sue=2012;` | | `jim=ann;` | | `sue=ann;` | |
| `ann = 24 hours;` | | `2011=ann;` | | `jim;` | |

***3. What is strange about the following pieces of code?***

Will they give a warning or error message? What warning or error will they give?

| | Problem? Error Message? | Corrected Code |
|---|---|---|
| `int main(void){`<br>`int jay;`<br>`double fred;`<br>`fred=6.2;`<br>`jay=fred;`<br>`return(0);`<br>`}` | | |
| `int main(void){`<br>`int jay;`<br>`long paul;`<br>`paul/=jay;`<br>`return(0);`<br>`}` | | |

NOTE: `paul/=jay;` means `paul = paul/jay;`

When you have completed the work, ask the TA to check your work. Be prepared to demonstrate compiling, linking and running the code.. Use the debugger if needed to demonstrate your ideas.

| Date | | Class ID | | | TA Stamp |
|---|---|---|---|---|---|
| Time | | Student ID | | | |
| Name | | | | | |

## 1.2 Fixed and Floating Point Calculations with Variables

Assume that you have a Base 10 calculator that can store 4 digits. Write the results of the calculations for fixed point, floating point and exact calculations. For fixed point, use 3 digits before the decimal point and 1 digit after. For floating point calculations use 3 figures in mantissa, and 1 figure in the exponent. Assume we do not store the sign.

| Equation | Fixed (Q3.1) | Floating (3+1) | "Exact" |
|---|---|---|---|
| 133 + 0.22 | | | |
| 133 – 0.99 | | | |
| 13.3 * 1.1 | | | |

## 1.3 Pointers

```
1:#include <stdio.h>
2:int main(void){
3: int fred, barney, wilma;
4: int *pebbles, *bambam;
5: int **dino;
6: fred = 1;
7: barney = 2;
8: wilma = 3;
9: pebbles = &fred;
10: bambam = &barney;
11: dino = &bambam;
12: continue;
13: return(0);
14:}
```

***1. Draw a VP diagram for the code.***

***2. What is the value of the following expressions?***

| a) *pebbles= | b) bambam= | c) *dino= | d) wilma + *pebbles = |
|---|---|---|---|

***3. What happens if we add the following at line 12?***

i) *bambam = *pebbles;
ii) bambam = pebbles;
iii) *dino = &wilma;

(Either give the answer or draw a VP diagram on the back of this page)

***4. Which assignments give errors or warnings? What message?***

a) fred = &pebbles;
b) pebbles = &fred;
c) pebbles = 5;
d) fred = 5;
e) dino = 5;
f) *dino = 5;
g) **dino = 5;
h) *dino = *pebbles;
i)*dino = pebbles;
j) *dino = &pebbles;

| Date | | Class ID | | | TA Stamp |
|---|---|---|---|---|---|
| Time | | Student ID | | | |
| Name | | | | | |

## 1.4 Pointers, Variables and Functions

Write a program that calls a function to calculate the volume of a sphere given its radius. Write two functions to do the same job. For one function, use "call-by-reference" for the required volume (and "call-by-value" for the radius) and let the other function just return the value of the volume. Use the following prototypes:

```
void volSphereCallByRef(double *vol, double r);
double volumeSphere(double r);
```

| Date | | Class ID | | | TA Stamp |
|---|---|---|---|---|---|
| Time | | Student ID | | | |
| Name | | | | | |

# 2 Arrays and Records

## 2.1 Convert this array version of strcat() to a pointer version

```
void strcat(char s[], char t[]){
  int i=0,j=0;
  while (s[i]) i++;
  while(t[j]){
    s[i] = t[j];
    j++;
    i++;
 }
  s[i]=0;
}
```

1. ***Type the above code into your computer***
2. ***Write a main() function for the testing of this function***
3. ***Modify the subroutine to make it all pointers***
4. ***Try to reduce the number of lines in the program.***

## 2.2 Arrays: Use an array to create a histogram of salaries for employees.

ABC insurance company pays its salespeople a basic salary of $100/week + commission of 10 percent of their gross sales for that week. For example, a salesperson that generates $2000 in policies in a week receives $100 plus 10 percent of $2000, for a total of $300. Write a C program that allows the user to enter the sales of 5 salespeople and outputs a histogram showing how many employees earned salaries in each of the following ranges:

- $100.00-$299.99
- $200.00-$299.99
- $300.00-$399.99
- $400.00-$499.99
- $500.00-$599.99
- $600.00-$699.99
- Over $700.00

The following is a sample user screen that you should try to generate.

```
Input salesperson[1] sales>>
Keyboard Input:  2000
…
Input salesperson[5] sales>>
Keyboard Input:  4500
      Salary           Number
      $100.00-$199.99   0
      $200.00-$299.99   1
…
      $600.00-$699.99   0
```

| Date | | Class ID | | | TA Stamp |
|---|---|---|---|---|---|
| Time | | Student ID | | | |
| Name | | | | | |

```
      Over $700.00        0
```

## 2.3 Records

***1. Enter the following code into your computer***

```
/* recTut.c :  to help us to learn record access */
#include <stdio.h>
struct Customer{
  char lastName[15];
  int id;
  struct{
    char city[15];
    char zipCode[6];
  } personal;
};
int main(void){
  struct Customer custPtr=NULL;
  struct Customer custRec= {"Ee",7,{"Malacca","75200" } };
  custPtr = &custRec;
/* place printf statements here */
  return(0);
}
```

***2. Add code to print each member twice: using custRec and using *custPtr.***

1. Member lastName
2. Member id
3. Member city of member personal
4. Member zipCode of member personal

***3. Modify "`struct Customer`" definition. Break apart the internal structure.***

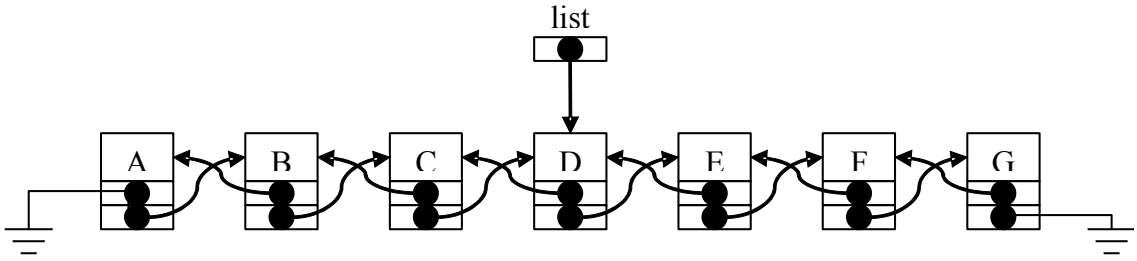
(i.e. eliminate `struct { ....}personal`)

| Date | | Class ID | | | TA Stamp |
|---|---|---|---|---|---|
| Time | | Student ID | | | |
| Name | | | | | |

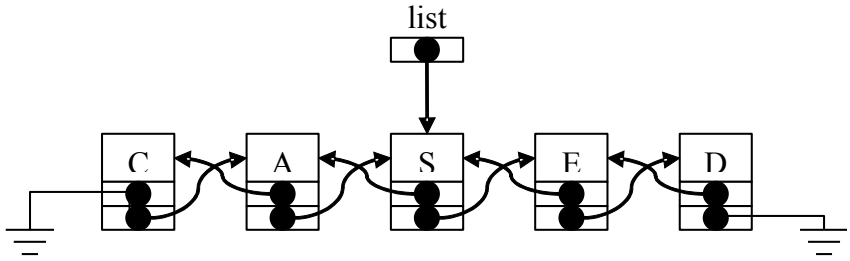
# 3 Linked Lists

## 3.1 Coding of a Doubly Linked List

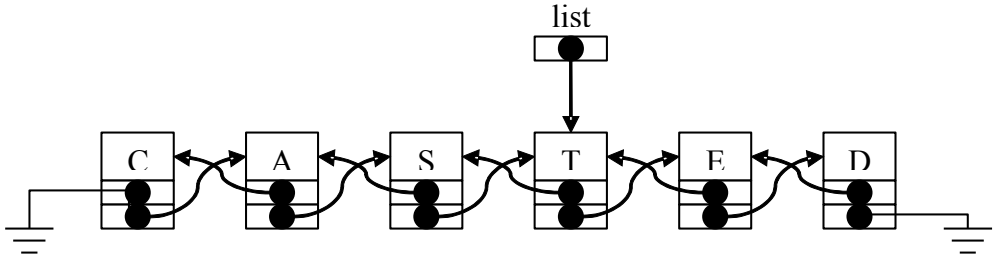
Given a doubly-linked list that looks like this:




In a linked list one needs to have a function to create a new node and connect it to the right of list. For example, if we have:




and we called insertLL(list,'T'), we would get



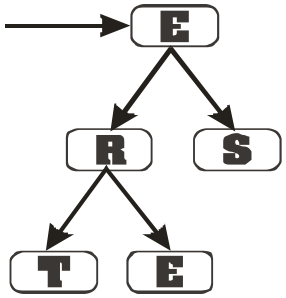

Note that the function needs to take care of the special cases of an empty list (i.e. list is pointing to NULL), and the special case of if list is pointing to the last node.

1. ***Use VP diagrams to show insertLL() inserting a node.***
2. ***Implement insertLL() in C, and try it out. Don't forget the special cases!***
3. ***Use VP diagrams to describe a function deleting a node. (Call it deleteLL().)***
4. ***Implement the node deleting function in ANSI-C***

| Date | | Class ID | | | TA Stamp |
|---|---|---|---|---|---|
| Time | | Student ID | | | |
| Name | | | | | |

# 4 Trees

***1. Traverse this tree in pre-, in-, and post-order:***




Pre-order:

In-order:

Post-Order:

***2. Design binary parse trees for the following arithmetic expressions.***

1. ANS = 20*3 – 60/4 + 2
2. A tree contains 3 number (“5”, “6”, “2”) as nodes along with the mathematical operations (i.e. “*”, “+” or “-“) Design a tree such that when the calculation is complete, the final answer is “9”.

***3. Define (in ANSI-C) a binary tree node containing a record.***

The record should contain a person' name and birth date. (Hint : Write a definition for the record, then use this record in the binary tree node)

| Date | | Class ID | | | TA Stamp |
|---|---|---|---|---|---|
| Time | | Student ID | | | |
| Name | | | | | |

# 5 Graphs and Sets

## 5.1 Graphs

***1. Draw the graph for the adjacency matrix***

| | l | o | v | e | d |
|---|---|---|---|---|---|
| l | 0 | 1 | 3 | 0 | 1 |
| o | 2 | 0 | 1 | 1 | INF |
| v | 1 | 1 | 0 | 0 | 2 |
| e | 3 | 1 | 0 | 0 | 0 |
| d | 1 | 0 | 0 | 1 | 0 |

***2. What type of graphs is this? (i.e. simple, di-, weighted)***

_____

***3. Rewrite the adjacency matrix as a list.***

| |
|---|
| |
| |
| |
| |
| |

***4. Traverse the graph in pre- and post-order (alphabetically)***

| | |
|---|---|
| pre-order: | |
| post-order: | |

| Date | | Class ID | | | TA Stamp |
|---|---|---|---|---|---|
| Time | | Student ID | | | |
| Name | | | | | |

# 6 Building Algorithms: Basic Techniques

## 6.1 Recursion: The Knapsack Problem

A thief robbing a safe finds it filled with items of varying size and value. Unfortunately, he only has a small knapsack to carry the loot. The problem is to find the combination of items that the robber should choose in order to maximize the total value of all the stolen items. For example, if the robber has a knapsack of `size=20` and the safe contains

| item | description | value (US$1000) | size |
|---|---|---|---|
| 1 | gold bar | 5 | 3 |
| 2 | silver bar | 5 | 9 |
| 3 | diamond ring | 3 | 1 |
| 4 | necklace | 2 | 4 |
| 5 | Chinese RMB | 4 | 3 |
| 6 | US$100 bills | 7 | 5 |
| 7 | Ancient China | 8 | 8 |
| 8 | gold broach | 4 | 4 |
| 9 | pendant | 1 | 1 |

The robber needs to decide what to take and what to leave behind.

In a recursive solution, each time we choose an item, we assume that we can find an optimal way to pack the rest of the knapsack.

1. ***Define a structure (named `Item`) to hold the name, value and size of an item.***
2. ***Write an ANSI-C function to read the items in the safe.***

   Read the data from a file. Store in an array of records. Name function `readItems()`
3. ***Rewrite this algorithm to maximize the value of items in the knapsack.***

   Rewrite the algorithm in UML and in ANSI-C .

```
int knapsack(int spaceRemainingInTheKnapsack){
 int totalValueOfItems=0;
 for(each item)
  if(there is room to place this item in the knapsack)
    Add it to the knapsack.
    Reduce the available space in knapsack by size of item.
    value = knapsack(spaceRemainingAfterIncludingThisItem)
          + ValueOfThisItem
    if(value > previousMaximumValue)
       Choose to take this item, increase the maximum value.
 return(maximum value);
```

4. ***Compile, Link and Run your Code***

| Date | | Class ID | | | TA Stamp |
|---|---|---|---|---|---|
| Time | | Student ID | | | |
| Name | | | | | |

# 7 Building Algorithms: Key Concepts

There are two ways to determine the time complexity of algorithms -- empirical analysis or by studying the implemented algorithm. In this lesson we will do both…

## 7.1 Time Complexity and Big-OH by Empirical Analysis

For this case we simply:

1. Choose the size of dataset (`n`)
2. Run the program for this data set & record time (`time`) taken by program to run.
3. Increase size of the data set (`n`).
4. GOTO 2
5. Fit a polynomial or combination of functions to your `time`=f(`n`)*

*(eg. could use Microsoft Excel to plot and help fit).

If the program is dependent on the nature of the input data (and not just the problem size), the above procedure must be repeated for a variety of data sets. This technique provides a good estimation of average time complexity for the program and requires no knowledge of the detailed inner workings of a program or algorithm.

In this first problem, we will estimate the average time complexity of a program for which we only have an executable version. Please download and run the program **timecomp.exe.** (Download the program from the tutorial section of www.sdba.info). Use the above algorithm to calculate this unknown program's asymptotic time complexity.

## 7.2 Calculate the time complexity and Big-OH of these sections of code

```
void ian(int n){
 int i;
 for (i=0; i<n; i++)
  printf("%d",i);
}
```

T(n)= ->Big Oh=

```
int betty(int n){
 if(n>1)
  return (betty(n-1)+n);
 else return(1);
}
```

T(n)= -->Big Oh=

```
void siewfong(int n){
 int i, j;
 for (i=0; i<n; i++)
  printf("%d",i);
 for (j=n; j>0; j--)
  printf("%d",j);
}
```

T(n)= -->Big Oh=

```
void twiddle(int n){
 int i, j;
 for (i=0; i<n; i++)
  for (j=n; j>0; j--)
   printf("%d %d",i,j);
}
```

T(n)=

```
void dino(int n){
 int i, j;
 for (i=0; i<n; i++)
  for (j=0; j<i; j++)
```

| Date | | Class ID | | | TA Stamp |
|---|---|---|---|---|---|
| Time | | Student ID | | | |
| Name | | | | | |

```
   printf("%d",j);
}
```

T(n) -->Big Oh=

| Date | | Class ID | | | TA Stamp |
|---|---|---|---|---|---|
| Time | | Student ID | | | |
| Name | | | | | |

# 8 Searching

## 8.1 Searching An Array By Hand

***1. Illustrate step-by-step a linear search for "k"***

| a | c | e | g | k | l | o | p | q | r | s | t | u | v | w | z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

***2. Illustrate step-by-step a binary search for "k"***

| a | c | e | g | k | l | o | p | q | r | s | t | u | v | w | z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

***3. Draw a (balanced) binary search tree for "k"***

You need only draw the part of the tree one searches on!

***4. Code a linear search function for a character's existence in a string***

Input: a string and character for which to search. Use the following calling routine, you supply the function `search()`. You may not use library functions from `<string.h>` to do the searching.

| Date | | Class ID | | | TA Stamp |
|---|---|---|---|---|---|
| Time | | Student ID | | | |
| Name | | | | | |

```
/* search.c : search for a char in a string */
/* (c)2001 by J D White */
#include <stdio.h>
int search(char *a, char key);
int main(void){
  char key;
  char a[]={"I hope to live in Taiwan, God willing."};
  key=getchar();
  if(search(a,key))puts("found);
  else puts("Not found");
}
```

## 8.2 Hash Tables

### *1. Place the 7 keys in a hash table of size 9.*

Use the modular arithmetic method where "index = key mode size". Resolve collisions by linear probing. Remember that the ASCII value of 'a' is 97;

| Key | a | m | g | o | r | i | t | | |
|---|---|---|---|---|---|---|---|---|---|
| Value | 97 | | | | | | | | |
| Modulus | 7 | | | | | | | | |
| index/step | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| 1 | | | | | | | | a | |
| 2 | | | | | | | | | |
| 3 | | | | | | | | | |
| 4 | | | | | | | | | |
| 5 | | | | | | | | | |
| 6 | | | | | | | | | |
| 7 | | | | | | | | | |

| Date | | Class ID | | | TA Stamp |
|---|---|---|---|---|---|
| Time | | Student ID | | | |
| Name | | | | | |

# 9 Sorting

## 9.1 Basic Techniques

### *1. Selection*

| Step\Key | a | l | g | o | r | i | t | h | m |
|---|---|---|---|---|---|---|---|---|---|
| 1 | | | | | | | | | |
| 2 | | | | | | | | | |
| 3 | | | | | | | | | |
| 4 | | | | | | | | | |
| 5 | | | | | | | | | |
| 6 | | | | | | | | | |
| 7 | | | | | | | | | |
| 8 | | | | | | | | | |
| 9 | | | | | | | | | |

### *2. Insertion (square brackets around inserted character)*

| Key | a | l | g | o | r | i | t | h | m |
|---|---|---|---|---|---|---|---|---|---|
| 1 | | | | | | | | | |
| 2 | | | | | | | | | |
| 3 | | | | | | | | | |
| 4 | | | | | | | | | |
| 5 | | | | | | | | | |
| 6 | | | | | | | | | |
| 7 | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |

## 9.2 Advanced Techniques

### *1. Merge (add dividers)*

| Key | a | l | g | o | r | i | t | h | m |
|---|---|---|---|---|---|---|---|---|---|
| 1 | | | | | | | | | |
| 2 | | | | | | | | | |
| 3 | | | | | | | | | |
| 4 | | | | | | | | | |

| Date | | Class ID | | | TA Stamp |
|---|---|---|---|---|---|
| Time | | Student ID | | | |
| Name | | | | | |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 5 | | | | | | | | | |
| 6 | | | | | | | | | |
| 7 | | | | | | | | | |
| 8 | | | | | | | | | |

### *2. Quick sort*

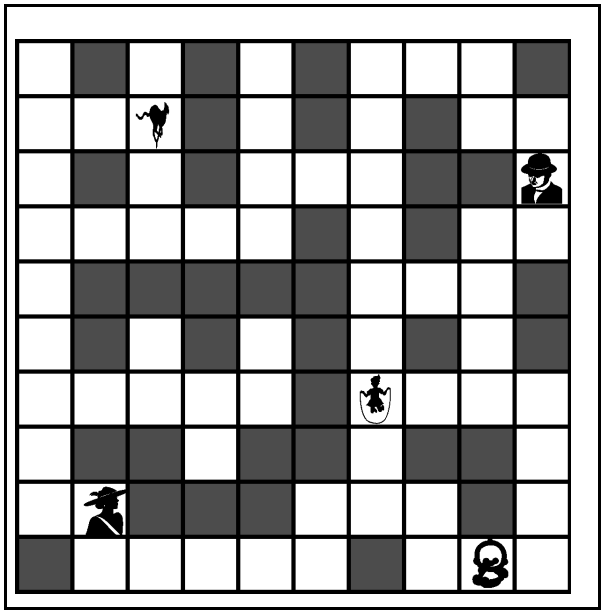
(square brackets around pivot, curved brackets for elements already sorted)

| Key | a | l | g | o | r | i | t | h | m |
|---|---|---|---|---|---|---|---|---|---|
| 1 | | | | | | | | | |
| 2 | | | | | | | | | |
| 3 | | | | | | | | | |
| 4 | | | | | | | | | |
| 5 | | | | | | | | | |
| 6 | | | | | | | | | |
| 7 | | | | | | | | | |
| 8 | | | | | | | | | |
| 9 | | | | | | | | | |
| 10 | | | | | | | | | |
| 11 | | | | | | | | | |
| 12 | | | | | | | | | |
| 13 | | | | | | | | | |
| 14 | | | | | | | | | |
| 15 | | | | | | | | | |
| 16 | | | | | | | | | |
| 17 | | | | | | | | | |

| Date | | Class ID | | | TA Stamp |
|---|---|---|---|---|---|
| Time | | Student ID | | | |
| Name | | | | | |

# 10 NP-Hard Problems

## 10.1 Help Xiao-Ming find his wife, Xiu-Man, in this maze.



1. ***Use the backtracking algorithm "maze" to find Xiu-man.***
2. ***Move through the maze using the greedy algorithm discussed in class.***
   (Assume Xiao-ming can hear his wife shouting)
3. ***Move through the maze using your own algorithm or heuristic.***
4. ***Compare the solutions for your 3 algorithms. Which is shortest?***

| |
|---|
| |

5. ***Determine the shortest path that Xiaoming can travel to reunite his whole family***
   (Xiu-man, Ai-en, You-en and Lightening the cat.) Use Dijkstra's algorithm

| Date | | Class ID | | | TA Stamp |
|---|---|---|---|---|---|
| Time | | Student ID | | | |
| Name | | | | | |

## 10.2 Dijkstra Algorithm

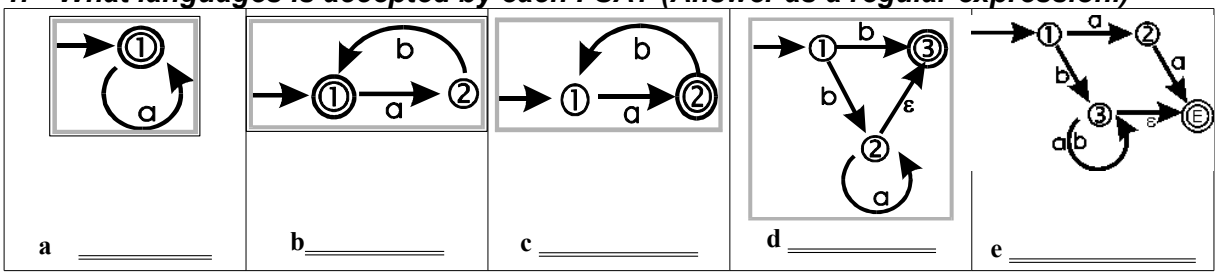
1. ***Chapter 10 in textbook, question 2***
2. ***Chapter 10 in textbook question 3***
3. ***Create a minimum spanning tree for graph LOVED.***

Use Dijkstra's algorithm. Start at D and find the shortest distance to each of the other nodes. (Find this graph in the tutorial on graphs).

| Date | | Class ID | | | TA Stamp |
|---|---|---|---|---|---|
| Time | | Student ID | | | |
| Name | | | | | |

# 11 Finite State Automata

***1. What languages is accepted by each FSA? (Answer as a regular expression.)***




**a** _____ **b**_____ **c** _____ **d** _____ **e** _____

***2. Which one of the above FSAs is deterministic (i.e. DFSA)?***

_____

***3. Draw a DFSA for each regular expression. Convert to NFSA.***

| RE | Σ | DFSA | NFSA |
|---|---|---|---|
| **a*b*c** | **a,b,c** | | |
| **(a \| b)*** | **a,b** | | |
| **(a*\|b*)** | **a,b** | | |
| **(a*b)\|(c+d)** | **a,b,c,d** | | |

| Date | | Class ID | | | TA Stamp |
|---|---|---|---|---|---|
| Time | | Student ID | | | |
| Name | | | | | |

# 12 Turing Machines

### *1. Run Turing Machines*

- Download the TM Simulator from the tutorial or theory section of the SDBA website.
- Run the three machines with their respective tapes. Try to follow what is happening.
- Modify the tapes and observe what happens.
- Modify the Turing Machine programs and observe what happens.

### *2. Run this TM with given input tape. Write final tape. Draw a FSA controller.*

| TM Program | Tape (input → final) | FSA Controller |
|---|---|---|
| ;add1.tm:add 1 to binary number<br>(MR,0,0,>,MR) ; Move Right<br>(MR,1,1,>,MR)<br>(MR, , ,<,ADD)<br>(ADD,0,1,>,ML) ; ADD one<br>(ADD, ,1,>,ML)<br>(ADD,1,0,<,ADD)<br>(ML,0,0,<,ML) ; Move Left<br>(ML,1,1,<,ML)<br>(ML, , , ,STOP) | 110 →<br>1 →<br>0 →<br>111 → | |

### *3. Write a TM control file (.tm) to subtract 1 from a binary number.*

Test run with TM Simulator. Draw the controller using an FSA-like state diagram.

| Controller | Test Tape (input → final) |
|---|---|
| | 1 → 0 |
| | 11 → 10 |
| | 10 → 01 |
| | FSA Controller |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |

| Date | | Class ID | | | TA Stamp |
|---|---|---|---|---|---|
| Time | | Student ID | | | |
| Name | | | | | |

### *4. Write a TM control file (.tm) to add 2 to a binary number.*

Test run with TM Simulator. Draw the controller using an FSA-like state diagram.

| Controller | Test Tape (input → final) |
|---|---|
| | 0 → 10 |
| | 1 → 11 |
| | 100 → 110 |
| | **FSA Controller** |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |

| Date | | Class ID | | | TA Stamp |
|---|---|---|---|---|---|
| Time | | Student ID | | | |
| Name | | | | | |

# 13 Contents of Weekly Tutorials

## 13.1 Yuan Ze University (981)

The following list includes the tutorial number used at YZU and the contents of the tutorial.

1. Chapter 0
2. Chapter 1 Questions 1.2 and 1.3
3. Chapter 2 Questions 2.1 and 2.2
4. Chapter 2 Questions 2.3
5. Chapter 3 Questions 3.1
6. Chapter 6 Questions 6.1
7. Chapter 7 Questionss 7.1 and 7.2
8. Midterm Paper Discussion
9. Chapter 4 Questions 4.1,4.2,4.3 and Chapter 8 Question 8.1.4
10. Chapter 8 Questions 8.1.1, 8.1.2, 8.1.3 and 8.2
11. Chapter 9 Questions 9.1.1, 9.1.2 and 9.2.1
12. Chapter 9 Questions 9.2.2 and Chapter 5 Questions 5.1
13. Chapter 10 Questions 10.1
14. Chapter 10 Questions 10.2

## 13.2 Multimedia University