

Section 10.1

1) Describe the difference between sexual reproduction in diploid organisms, binary fission in haploid organisms, and fusion in haploid organisms.

In diploid organisms, each parent produces a haploid gamete, and the two gametes unite to produce a diploid zygote, which then grows into an adult. The zygote has half of the genome from each parent.

Some haploid organisms (unicellular) reproduce through fusion: two parent cells combine to produce a transient meiocyte (diploid), which then undergoes meiosis, producing four haploid descendant cells. Each descendant has a mixture of genes from each parent, which makes this type of reproduction sexual.

Most haploid organisms reproduce asexually, through fusion: one parent cell splits into two copies with identical genomes.

2) Suppose a diploid organism has 10 chromosomes in one of its genomes.

(a) How many chromosomes are in each of its somatic cells?

Somatic cells are diploid, therefore they have two genomes, or 20 chromosomes.

(b) How many chromosomes are in each of its gametes?

Gametes are haploid, therefore they have only one genome, or 10 chromosomes.

3. Suppose two adult haploid organisms reproduce by fusion.

(a) How many children will be produced?

Four.

(b) Will the genetic content of the children all be the same?

No, in general each of the four children is genetically distinct (each inherits different combinations of genes from each parent).

4. Consider the eye color of a human being as determined by the *bey2* gene. Recall that the allele for brown eyes is dominant. For each of the following parent allele combinations, determine the eye color of the individual.

Father	Mother	Child
BLUE	BLUE	BLUE
BLUE	BROWN	BROWN
BROWN	BLUE	BROWN
BROWN	BROWN	BROWN

Section 10.2

5) Consider Table 10.1. Suppose the fitnesses of the 8 individuals are .61, .23, .85, .11, .27, .36, .55, and .44. Compute the normed fitnesses and the cumulative normed fitnesses.

Fitness	Normed fitness	Cumulative normed fitness
.61	.178	.178
.23	.067	.245
.85	.249	.494
.11	.032	.526
.27	.079	.605
.36	.105	.710
.55	.161	.871
.44	.129	1.000

Sum 3.42

6) Suppose we perform basic crossover as illustrated in Table 10.3, the parents are 01101110 and 11010101, and the starting and ending points for crossover are 3 and 7. Show the two children produced.

We assume that the points for crossover are numbered from left to right, from 1 to 8.

The children are: 0101 0100
 1110 1111

7) Implement the genetic algorithm for finding the value of x that maximizes $f(x) = \sin(x\pi/256)$, which is discussed in Section 10.2.2.

Implementations will vary, however, in C/C++ efficient mutation and crossover operations should take advantage of the bitwise operators. Below is a short example showing how the bit in position x can be swapped (crossover) between two 8-bit numbers a and b . The bit positions are 0 on the right and 7 on the left.

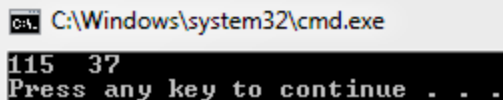
```
#include <iostream>
using namespace std;

int main(){
    unsigned char a, b, mask, temp_a, temp_b, x;
    a = 99;
    b = 53;
    x = 4;      //swap bit 4 (numbered 0 through 7, R to L)
    mask = 1;
    mask = mask <<x;
    temp_a = mask & a;
    temp_b = mask & b;

    mask = ~mask;  //reverse mask
    a = a & mask;
    b = b & mask;
    a = a | temp_b;    //115
    b = b | temp_a;    // 37

    cout <<(int)a <<" " <<(int)b <<endl;

    return 0;
}
```

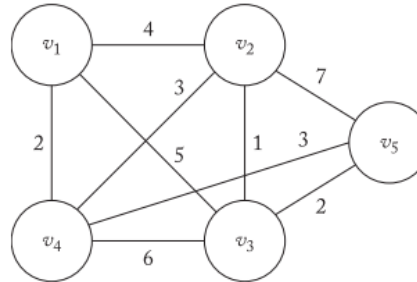


```
C:\Windows\system32\cmd.exe
115 37
Press any key to continue . . .
```

To swap an entire crossover sequence, a mask with consecutive ones should be constructed first, and then the process is similar.

8) Consider the instance of the TSP in Figure 10.12. Assume the weights in both directions on an edge are the same. Find the shortest tour.

Figure 10.12
An instance
of TSP.



We use the implementation from Exercise 6-10, with a 5-by-5 matrix:

```
#include <iostream>
using namespace std;
#define n 5
int W[n+1][n+1] = { {0, 0, 0, 0, 0, 0},
                    {0, 0, 4, 5, 2, 0},
                    {0, 4, 0, 1, 3, 7},
                    {0, 5, 1, 0, 6, 2},
                    {0, 2, 3, 6, 0, 3},
                    {0, 0, 7, 2, 3, 0} };

int best_vindex[n] = {-1};
int vindex[n] = {-1};
int minimum = 1000;

bool promising(int i){
    int j;
    bool switchie;
    if (i==n-1 && !W[vindex[n-1]][vindex[0]]) //No edge to close circuit
        switchie = false;
    else if (i>0 && !W[vindex[i-1]][vindex[i]]) //No edge to current node
        switchie = false;
    else{ //Check if vertex has already been selected
        switchie = true;
        j = 1;
        while (j<i && switchie){
            if (vindex[i] == vindex[j])
                switchie = false;
            j++;
        }
    }
    return switchie;
}

int evaluate(int vindex[]){
    int sum = 0;
    for (int i=0; i<n-1; i++){
        sum += W[vindex[i]][vindex[i+1]];
    }
    sum += W[vindex[n-1]][vindex[0]]; //closing the circuit
    return sum;
}

void hamiltonian(int i) {
    int j;
    if(promising(i)) {
        if (i == n-1){
            int eva = evaluate(vindex);
            if (eva <= minimum) {
```

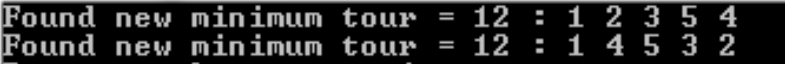
```

        minimum = eva;
        cout <<"Found new minimum tour = " <<minimum <<" : ";
        for (int k=0; k<n; k++){
            cout << vindex[k] <<" ";
            best_vindex[k] = vindex[k]; //update min. circuit
        }
        cout <<endl;
    }
    else
        for(j=2; j<=n; j++) {
            vindex[i+1] = j;
            hamiltonian(i+1);
        }
    }
}

int main(){
    vindex[0] = 1;
    hamiltonian(0);

    return 0;
}

```



```

Found new minimum tour = 12 : 1 2 3 5 4
Found new minimum tour = 12 : 1 4 5 3 2

```

As expected, there are two minimal tours, one being the reverse of the other, since all edges are bi-directional (undirected graph).

9) Suppose we perform order crossover, the parents are 3 5 2 1 4 6 8 7 9 and 5 3 2 6 9 1 8 7 4, and the starting and ending points for the pick are 4 and 7. Show the two children produced.

We assume that the points for crossover are numbered from left to right, from 1 to 8.

The children are: 9 7 5 1 4 6 8 3 2
 4 7 3 6 9 1 8 5 2 The picks are shaded.

10) Consider the instance of the TSP in Figure 10.12. Apply the Nearest Neighbor Algorithm starting with each of the vertices. Do any of them yield the shortest tour?

The only two starting vertices that yield tours are v2 and v4:

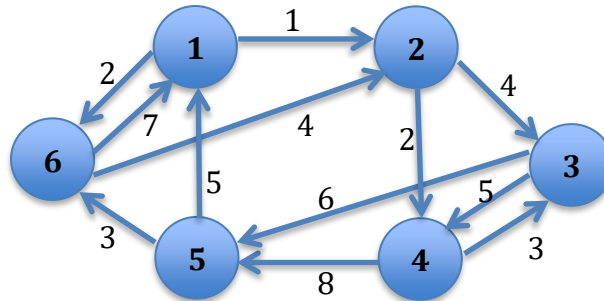
v2 → v3 → v5 → v4 → v1 → v2 Length is 12

v4 → v1 → v2 → v3 → v5 → v4 Length is 12

They are both the same tour (after a circular permutation), namely the first shortest tour found in Exercise 8.

11) Form the union graph of the two tours shown in Figure 10.13, and apply the Nearest Neighbor Algorithm to the resultant graph starting at vertex v_5 .

The union graph is:



The sequence of nodes starting at v_5 is: $v_5 \rightarrow v_6 \rightarrow v_2 \rightarrow v_4 \rightarrow v_3 \rightarrow \text{STOP}$, since there is no edge to the remaining node v_1 .

If, however, we start at v_1 , we obtain the tour $v_1 \rightarrow v_2 \rightarrow v_4 \rightarrow v_3 \rightarrow v_5 \rightarrow v_6 \rightarrow v_1$, of cost 22.

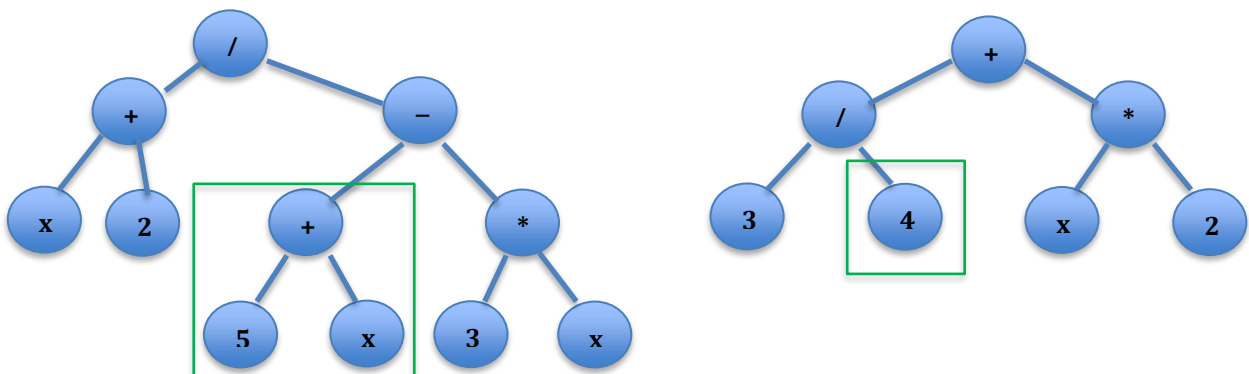
12) Apply the Greedy Edge Algorithm to the instance of the TSP in Figure 10.12. Does it yield a shortest tour?

The first three shortest edges are 2-3 (cost 1), 1-4 (cost 2), and 3-5 (cost 2). They can all be added without violating the conditions for a tour. The next two edges are 2-4 and 4-5, both of cost 3, but they cannot be both added, because this would create a cycle composed of only the nodes $2 \rightarrow 3 \rightarrow 5 \rightarrow 4$. If we try to add 2-4, no tour can be found, but if we add 4-5, we obtain a tour after the addition of the next edge, 1-2 (cost 4).

The final tour is $1 \rightarrow 2 \rightarrow 3 \rightarrow 5 \rightarrow 4 \rightarrow 1$, of cost 12.

Section 10.3

13) Consider the two trees in Figure 10.8. Show the new trees that would be obtained if we exchanged the subtree starting with the “4” in the left tree with the subtree starting with the “+” in the right tree.



14) Consider the individual (program) in Figure 10.10. Show the moves produced by that program when negotiating the Santa Fe trail for the first 10 time steps.

Step nr.	Actions	End position and orientation
1	Move (and eat)	(1,2)→
2	Move (and eat)	(1,3)→
3	Move (and eat)	(1,4)→
4	L, R, R, L, R, move (and eat), move (and eat)	(3,4)↓
5	Move (and eat)	(4,4)↓
6	Move (and eat)	(5,4)↓
7	Move (and eat)	(6,4)↓
8	L, move (and eat), R, L, R, L, move (and eat)	(6,6)→
9	Move (and eat)	(6,7)→
10	L, R, R, L, R, L, move (not eat)	(6,8)→

15) Implement the genetic programming algorithm for the Santa Fe trail discussed in Section 10.3.2.

Implementations will vary.