**1. True or False Qusetions (2×5=10 points)**

(1) Monte Carlo algorithm is always correct ( False )

(2) The language A = {w∈{0,1}\* | there are same number of appearance of 01 and 10} is regular ( True )

**2. Fill blanks(3×10=30 points)**

(1) A divide-and-conquer paradigm involves **divide, conquer, combine**

(2) Turing decidable languages are **close** (close or not close) under the operation concatenation

3. **(10 points)**The rod-cutting problem. Given a rod of length n inches and the price p[i] of selling a rod of length i, where i = 1,2,…,n. Determine the maximum revenue rn obtainable by cutting up the rod and selling the pieces.

Input: n and an array p of length n, where p is the sequence of prices.

Output: The optimal revenue and an optimal solution.

Write a psudo-code solving the problem, and write the computation process of your psudo-code for input: n=5 and prices p[1]=1, p[2]=5, p[3]=8, p[4]=9, p[5]=10.

Sol. The psudo-code

//p[i]: price of length i rod

//r[k]: optimal revenue for a length k rod

//r[k] = max1≤i≤k p[i] + r[k-i], r[0] = 0

Extended-bottom-up-cut-rod(p,n)

1. r[0]=0

2. for k = 1 to n

3. r[k] = p[k], s[k]=k

4. for i = k-1 to 1

5. if r[k] < p[i]+r[k-i],

6. r[k] = p[i]+r[k-i], s[k] = i

7. return r and s //return arrays

print-cut-rod(p,n)

1. (r,s)=Extended-bottom-up-cut-rod(p,n)

2. While n>0

3. print s[n]

4. n=n-s[n]

r[0]=0.

k=1

r[1]=p[1]=1, s[1]=1.

k=2

r[2]=p[2]=5, s[2]=2.

i=1: r[2]>p[1]+r[1]=2.

k=3

r[3]=p[3]=8, s[3]=3.

i=2: r[3]>p[2]+r[1]=6;

i=1: r[3]>p[1]+r[2]=6.

k=4

r[4]=p[4]=9, s[4]=4.

i=3:r[4]=p[3]+r[1]=9;

i=2:r[4]<p[2]+r[2]=10,r[4]=10,s[4]=2;

i=1:r[4]>p[1]+r[3]=9.

k=5

r[5]=p[5]=10, s[5]=5.

i=4: r[5]=p[4]+r[1]=10;

i=3: r[5]<p[3]+r[2]=13,r[5]=13,s[5]=3;

i=2: r[5]=p[2]+r[3]=13.

i=1: r[5]>p[1]+r[4]=11.

print r[5]=13

print s[5]=3

n=5-s[5]=2

print s[2]=2

Note: 5 points for the algorithm, 5 points for computation of example

4. **(15分)** Let Double-SAT = { <φ> | φ has at least two satisfying assignments }. We know that SAT is NP complete. Show that Double-SAT is NP complete.

Proof

(1) Show that Double-SAT∈NP

Construct the following NTM

N=“On input <φ>, φ is Boolean formula,

(a) non-deterministically generate two different assignments S,T.

(b) if φ=1 on both assignment S and T, then accept；otherwise，reject.”

Since the language of N is Double-SAT, and N runs in polynomial time so DOUBLE-SAT∈NP.

(2) Show that SAT ≤P DOUBLE-SAT.

Take any Boolean formula φ, add a new Boolean variable a, define

f(φ) = φ ∧ (a∨¬a)

Firstly, f can be computed in polynomial time.

Secondly, f is a mapping reduction from SAT to DOUBLE-SAT, i.e.

φ is satisfiable ⇔ f(φ) has two assignments

If φ has a satisfying assignment S, then

f(φ)=1 on the assignment S and a=1; f(φ)=1 on the assignment S and a=0.

f(φ) has at least two satisfying assignments.

If f(φ) has two satisfying assignments, then φ is satisfiable .

By (1) , (2) and SAT is NP complete, Double-SAT is NP complete.

Note:

(1) proof of NP, 5 points；

(2) proof of polynomial time reducible.

mapping, 3 points；

show that the mapping is polynomial computable, 1 points；

show that the mapping is a reduction, if part 2 points, only if part 2 points

(3) state the process, 2 points