Homework 1

Sunday, January 20, 2019 1:39 PM

1.) Nuts[1...n]
Bolts[1...n]
Already Sorted
n = size of Both Bolts and Nuts arrays

```
i = 0        // Nuts iterator
j = 0        // Bolts iterator
while (i < n and j < n)
        if Nuts[i] = Bolts[j]
                return true
        if Nuts[i] > Bolts[j]
                j++;
        if Nuts[j] < Bolts[i]
                i++;
```

2.)

A) A[1...n] A.size = N
t > 0

```
i = 0
j = n
while (i < j)
    if A[i] + A[j] = t
            return true
    if A[i] + A[j] < t
            i++;
    if A[i] + A[j] > t
            j++;
```

B) Sort A (nlgn)

```
for i = 0 to n
x = i + 1
y = n - 1
While (x < y)
    if (A[x] + A[y] + A[i] = t
            return true
    if (A[x] + A[y] + A[i] < t
            x++;
    if (A[x] + A[y] + A[i] > t
            y--
```

So j++

3.) A[1...n] unsorted

Sum = 1000

Sample Sum = 15

2	9	12	13	23	1	3	8	20

i

Set 2, 9, 12

temp: 15-2 = 13 13>0 hash doesn't have 13 Set.add(2)

temp = 15-9 = 6 6>0 Set doesn't have 6 Set.add(9)

temp = 15-12 = 3 3>0 set doesn't have 3 Set.add(12)

temp = 15-13 = 2 2>0 Set has 2 so return

```
Hashset Set ;
for i:0 to A.size
   temp = sum - A[i]
   if temp < 0
        continue
   if Set.contains(temp)
        return true
   Set.add(A[i])
```

NOTE: This only works if hashset.add and hashset.contains are really O(1) like the docs say

4.) A[1...n]

A[i] contains (x_i, y_i) of a point P_i

(1,2)	(2,4)	(3,7)	(1,2)	(7,12)	(1,9)

temp = Array.Sort() ⟶ assume heapsort w/ O(nlgn)

```
i:0
j:1
while(i < n-1 and j < n)
   if( i.x == j.x and i.y == j.y)
        return
   if(i.x != j.x)
        continue
```

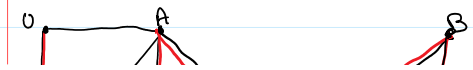
5.) 34.2-6

HAM-PATH = {(G, u, v)}

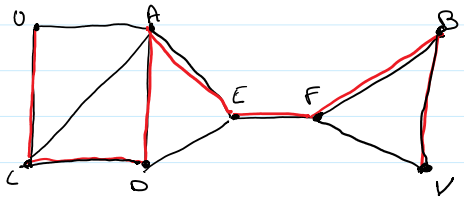
Hamiltonian Path

- Must use all verticies
- Can only hit any verter 1 time

Sample



Sample



Certificate (U, C, D, A, E, F, B, V)

$y = \{U, C, D, A, E, F, B, V\}$; y.length = n

X: 1.) Does the certificate have > 0 vertices
2.) Are there any repeated vertices
3.) Are all vertices used
4) Does the path start at U and end at V

Verification

1.) X=yes y does return yes
2. Runs Polynomial Time
1) O(n)
2.) O(n)
3.) O(n)
4.) O(1)

Total runtime is O(n)

Verification is satisfied so HAM-PATH is NP

6.) 34.4-5

Disjunctive Normal Form

- Boolean formula containing clauses of ANDs OR'd together
- Satisfiable if at least 1 conjunctive is satisfiable and equals 1
- if the clause contains both the literal and its negation it evaluates to 0
- Example

$A = (x_1 \wedge x_2 \wedge x_3) \vee (\bar{x}_1 \wedge x_2) \vee (x_1 \wedge \bar{x}_3) \vee (x_1 \wedge \bar{x}_1)$

$x_1 = 1$

$x_2 = 1$

$x_3 = 1$

$A = (1 \wedge 1 \wedge 1) \vee (0 \wedge 1) \vee (1 \wedge 0) \vee (1 \wedge 0)$

$A = (1) \vee (0) \vee (0) \vee (0) = 1$

Satisfiability Formula

1.) Loop through all clauses
2.) Loop through all literals in each clause

and evaluate the answer

- if the clause contains the literal and its negation return false
- if clause evaluates to True return true

```
n= # of clauses
m= # of literals in n[:]
for i = 0 to n
        value = true
    for j=0 to m
        value = value AND n[j]
    if value = true
        return true
return false
```

Running Time for the nested loop is
O(n) for the outter loop of clause
O(m) for the inner loop of literals
O(n * m)

7) 34.4-6

SAT = {⟨ϕ⟩ : ϕ is a satisfiable boolean formula}
ϕ has {$x_1, x_2, \dots x_n$} n literals
F is the given algorithm
A is an array of size n

```
if F(ϕ) is not satisfiable return false
  For i:0 to n
     BF = ϕ
     replace Xi in BF with 1
     if F(BF) == true
        A[i] = 1
     else
        A[i] = 0
        replace Xi in ϕ with A[i]
  return A
```

F runs in polynomial time
The function above loops n times calling F
so it also runs in polynomial time