[redacted]

ıdemic

CODE.
aking the rules is
f them. Understand
onstitute academic
oid them.

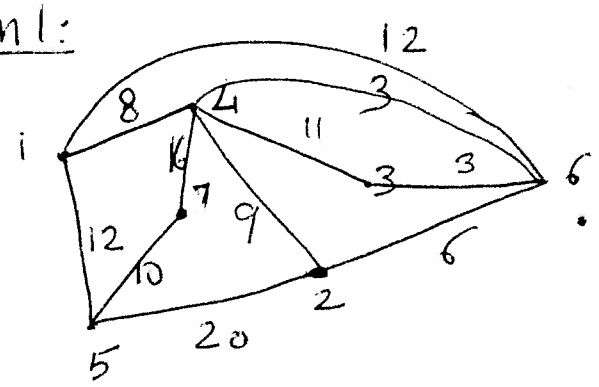
).
uation where you
t. Procrastination
speration -- and this
quate preparation
te understanding of
dequate sources of
yourself enough
gnment.

HEN YOU

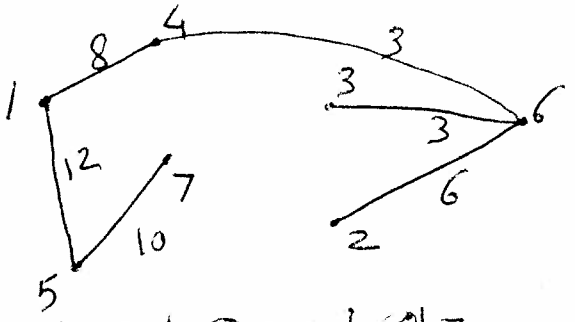
with work, don't
e several offices on
ou, including the
nter, the Writing
ence Desk and tutors
ment.

UBT, ASK.
professor for
s the one who can
dishonesty and what
by doing something
Academic Integrity
ed if you have

Problem 1:

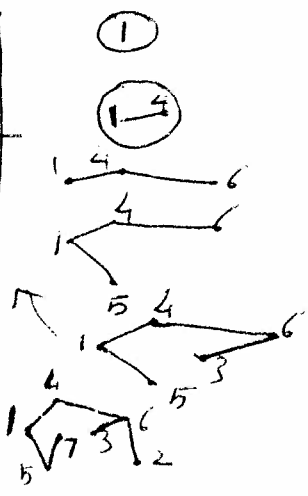


a)



b)

i	1	2	3	4	5	6	7
DIST[1]	0	∞	∞	(8)	12	12	∞
DIST[4]	0	17	19	8	12	(11)	24
DIST[6]	0	17	14	8	(12)	11	24
DIST[5]	0	17	(14)	8	12	11	22
DIST[3]	0	(17)	14	8	12	11	22
DIST[2]	0	17	14	8	12	11	(22)
DIST[7]	0	17	14	8	12	11	22



problem 2:

a)

x:	1	1	5	5	5	7	13	13
y:	2	2	3	3	3	1	2	2

b)

```
proc computey (in: x[1:n]; out: y[1:n])
{
    if n = 1 then
        y[1] = 1; return;
    endif
    computey (x[1:n/2], y[1:n/2]);
    computey (x[n/2+1:n], y[n/2+1:n]);
    if (x[n/2] ≠ x[n/2+1]) then
        return; // done
    else
        z = y[n/2] + y[n/2+1];
        for (i = n/2; x[i] == x[n/2]; i--)
            y[i] = z;
        for (i = n/2+1; x[i] == x[n/2]; i++)
            y[i] = z;
    endif
}
```

Worst-case: $O(n)$

$T(n) = 2T(n/2) + cn$
$\Rightarrow T(n) = O(n \log n)$

---

problem 3:

a) main idea:
1) sort the array $A[1:n]$ into $x[1:n]$
2) call computey $(x[1:n], y[1:n])$ of previous ~~method~~ problem
3) find the largest value in $y[1:n]$; say it is $y[m]$
4) return $x[m]$.

```
function maxfreq ( A[1:n])
{
    Mergesort( A[1:n], x[1:n]);          <- O(n log n)
    computey( x[1:n], y[1:n])            <- O(n log n)
    datatype M = y[1]; int m = 1
    for (i = 2; i <= n; i++)             |
        if y[i] > M then                 | O(n)
            M = y[i]; m = i;             |
        endif                            |
    endfor                               |
    return x[m];
}
```

b) time: $T(n) = O(n \log n) + O(n \log n) + O(n)$

$T(n) = O(n \log n)$

problem 4:

a) proc jobassign(in $C[1:n, 1:n]$; out $P[1:n]$)

{ // $P[i]$ ≡ the person assigned to job i

Boolean $A[1:n] = [false, false, \ldots, false]$;

for $i = 1$ to n do

*** find the best remaining person and assign him/her to job i;

best person is person j where

$C[i,j] = \min\{C[i,k] \mid A[k] = false\}$

**/

$M = +\infty$;

for $\ell = 1$ to n do

if ($A[\ell] = false$ and $C[i,\ell] < M$)

$M = C[i,\ell]$;

$j = \ell$;

endif

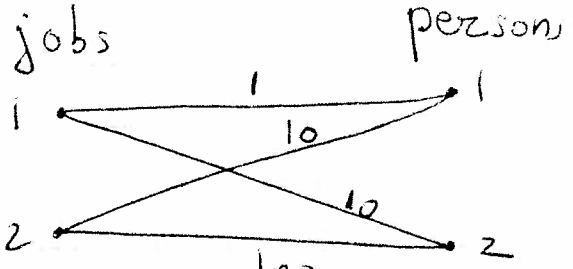
endfor $P[i] = j$; $A[j] = true$;

} endfor

$A[j]$ = true if person j has been assigned; false otherwise

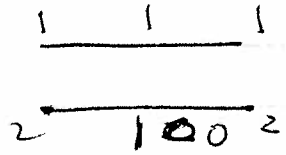
b) time: $T(n) = O(n^2)$.

c)



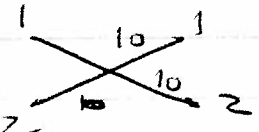
$c_{11} = 1$, $c_{12} = 10$, $c_{21} = 10$, $c_{22} = 100$

The greedy solution:



of cost = 101

An alternative solution:



has a cost = $10 + 10 = 20 < 101$.

$\Rightarrow$ the greedy solution is not optimal