Q1 (a)

***500/294 = 1.7006***

***500/203.84 = 2.4529***

***500/176.62 = 2.8309***

***500/164.92 = 3.0317***

Q1(b) Calculate the values for the Karp-Flatt metric. Furthermore, also interpret the results of Karp-Flatt metric and write your opinion accordingly.

***Screenshot_1***

***((1/1.70) – (1/2)) / (1 – 1/2) = 0.18***

***((1/2.45) – (1/4)) / (1 – 1/4) = 0.21***

***((1/2.83) – (1/6)) / (1 – 1/6) = 0.22***

***((1/3.03) – (1/8)) / (1 – 1/8) = 0.23***

**Since ‘e’ is increasing with p, it suggests that parallelization overhead is also contributing to the poor speedup. Hence, we need to reduce this overhead to improve speedups.**

**Q2(a)**

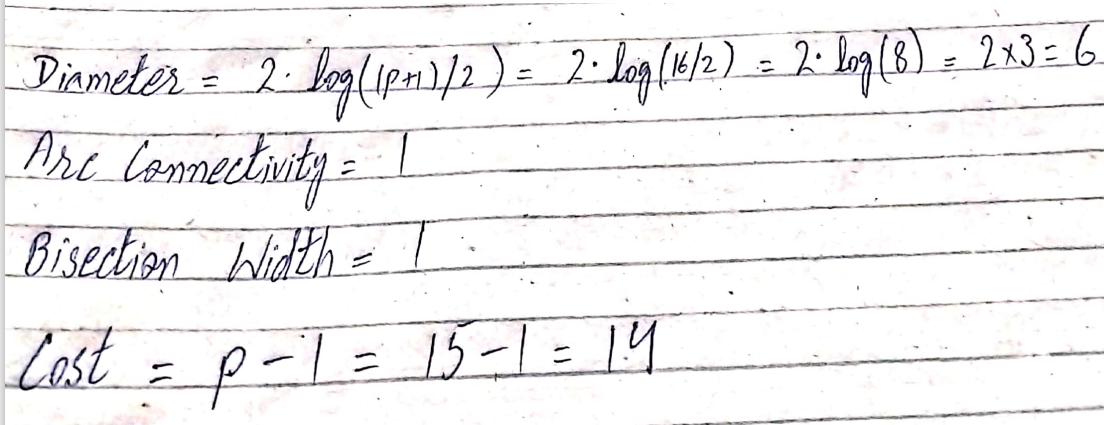
**Diameter = log2 (p) = log2 (16) = 4**

**Bisection width = p/2 = 16/2 = 8**

**Arc connectivity = log2 (p) = log2 (16) = 4**

**Cost = (p \* log2 (p) ) / 2 = (16 \* log2 (16)) / 2 = 32**

**Q2(b)**

****

**Q3: (a)** Maximum degree of concurrency = 4

(b) Total Amount of Work = 50

(c) Critical Path Length = 25

(d) Average degree of concurrency = 2

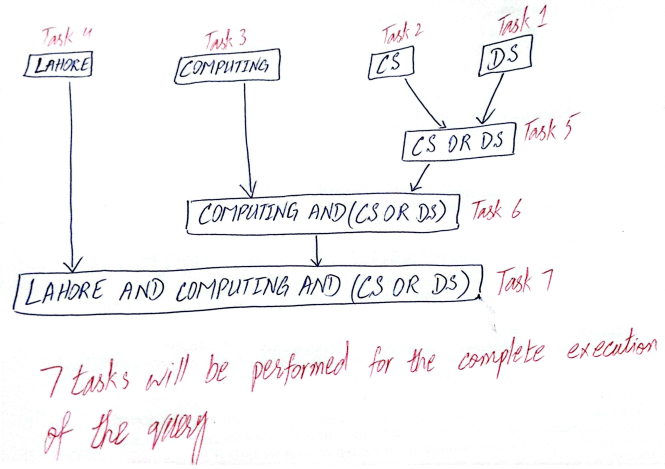
(e)

Tasks having corresponding weights 2, 8, 9, 6 may be mapped to Pa

Tasks having corresponding weights 7, 1 may be mapped to Pb

Tasks having corresponding weights 3, 4, 10 may be mapped to Pc

***Q4.***

****

***Q5 (a)***

static long num\_steps = 100000;

double step;

void main ()

{

int i; double x, total, sum = 0.0;

step = 1.0/(double) num\_steps;

for (i = 0; i < num\_steps; i++){

x = (i + 0.5) \* step;

sum = sum + 1.0/(1.0 + x + x \* x + x \* x \* x \* x);

}

total = step \* sum;

}

**Q5 (b)**

static long num\_steps = 100000;

double step;

#define NUM\_THREADS 4

int main(int argc, \_TCHAR\* argv[]) {

int i, nthreads;

double total, sum[NUM\_THREADS];

step = 1.0/(double) num\_steps;

omp\_set\_num\_threads(NUM\_THREADS);

#pragma omp parallel

{

int i, id, nthrds;

double x;

id = omp\_get\_thread\_num();

nthrds = omp\_get\_num\_threads();

if(id == 0)

nthreads = nthrds;

for (i=id, sum[id]=0.0; i< num\_steps; i=i+nthrds){

x = (i+0.5)\*step;

sum[id] += 1.0/(1.0 + x + x \* x + x \* x \* x \* x);

}

}

for(i=0, pi=0.0; i < nthreads; i++)

total += sum[i] \* step;

printf("%f\n", total);

return 0;

}