Task 1:

1b)

The runtimes for all the BFS algorithms including the one using cilk's internal work stealing mechanism.

Data sets	Sequential (seconds)	Parallel (seconds)	Parallel_deltafree (seconds)	Cilk's work stealing scheduler(seconds)
cage15	2166	1073	1139	718
cage14	574	305	324	192
freescale	878	597	513	186
wikipedia	1573	636	590	316
kkt-power	16	14	16	13
rmat100M	6754	2032	2074	1395
rmat1B	33644	6805	6790	6666

Task 2:

2a)

Let the graph be divided into n connected components - 1

For each connected component consisting of v_i vertices - Thus $\sum v_i = V$ - 2

Now for each connected component with - v_i vertices $v_i - 1$ edges can be be made part of Q without causing a cycle.

Thus number of edges in $Q = \sum v_i - 1 = |V| - n$ - 3

Minimum size of a connected component = 2

As there are no isolated vertices, Each vertex is part of a connected component .Thus maximum number of connected components possible - if all connected components have minimum size.

Thus n = $|V|/2$, hence Minimum number of edges in Q = $|V| - |V|/2$ = $|V|/2$

Hence Proved.

2b)

Consider $Q' \in Q$ as a set of edges comprising a connected component.

$|Q'|$ - denotes the number of edges in this set.

Thus number of vertices in Q' = $|Q'|$ + 1

Name those edges as E_i – for i going from 0 to $|Q'|$

Assume $Q'' \subseteq Q'$ such that it consists of all edges of Q' except the last edge $Q'th$

Thus $|Q''|$ = $|Q'| - 1$, similarly no. of vertices in Q''= $|Q'|$

If H[E_i] = 1 , then C[$E_i.u$] ≠ C[E_i.v] else if H[E_i] = 0 , then C[$E_i.u$] = C[E_i.v]

Thus for any connected component , if H[E_i] is known for all E_i , then knowing C[$E_0.u$]

, we can tell C[$E_{last}.v$]

Now consider for Q' and Q'' thus knowing C[$E_0.u$] and H[E_i] for - Q'' , if we were able to determine C[$E_{last}.v$] of Q' - that would mean that there is a cycle connecting some intermediate node of Q'' to last node of Q' - As cycles are not allowed - $C[E_{last}.v]$ of $Q'$is a completely independent of event and knowing all information about H[E_i] for Q'' , we cannot determine H[E_{last}] for Q'.

2c)

Both Random Hook and In class algorithm randomly assign Heads and Tails to all vertices and hooks an edge (u,v) if u = Tails and v =Heads

Random Hook uses reversing the head and tails maneuver in lines 9 to 15 in order to try and hook more vertices in a single iteration. This help in converging hooking faster.

Consider the worst case of n vertices 0 to n-1

With 0 to n-2 being heads and n-1th being tails – here not even a single hooking would occur in original algorithm but in random hook after flipping the heads and tails – hooking can be achieved in a single iteration of random hook .

Hence, Random Hooking is better than the original in class algorithm.

2d)

Let X_i be a random variable , which has value =1 if L value changes , else 0

E[X_i] = 1 * ¼ = ¼

Thus for | V | such vertices

E [X] = μ = |V| /4

Thus expected in |V|/4 but we want a bound on |V|/16

Thus (1 - δ) μ = |V|/16 , thus $\delta = 3/4$

Applying Chernov bound

Pr[X < (1 - δ)] ≤ $e^{-\frac{\delta^2 \mu}{2}}$

Pr[X < |V|/16] ≤ $e^{-\frac{3/4^2 * |V|/4}{2}}$

Pr[X < |V|/16] ≤ $e^{-\frac{3/4^2 * |V|/4}{2}}$

Pr[X < |V|/16] ≤ $e^{-\frac{|V|}{32}*(\frac{3}{2})^2}$

1 - Pr[X < |V|/16] > 1 - $e^{-\frac{|V|}{32}*(\frac{3}{2})^2}$

Pr[X > |V|/16] > 1 - $e^{-\frac{|V|}{32}*(\frac{3}{2})^2}$

Thus Pr[X > |V|/16] > 1 - $e^{-\frac{|V|}{32}*(\frac{3}{2})^2}$ > 1 - $e^{-\frac{|V|}{32}}$

2e)

Work for ParallelCC2-

Parallel CC2

1. line 1 – O(1)
2. associate the edge (u,v) with u and v (line2)- O(m)
3. Random hook(line 3)- O(n)
4. Construct V' (line 5)- O(n)
5. Construct E' (line 6)– O(m)
6. map the solution back to the current instance(line7)- O(n)

Recursion will execute steps 1-6 logn times

(From question 2d, we see that each call to random_hook() reduces the number of vertices by a fraction of 1/16. Therefore, the recursion depth will be logn and n=0 implies m=0 which is the base condition)

So complexity becomes- O((m+n)*logn)

Span for ParallelCC2-

Parallel CC2

1. line 1 – O(1)
2. associate the edge (u,v) with u and v (line2)- O(logm)
3. Random hook(line 3)- O(logn)
4. Construct V' (line 5) - O(n)
5. Construct E' (line 6)– O(m)
6. map the solution back to the current instance(line7)- O(logn)

Recursion will execute steps 1-6 logn times

So complexity becomes- O((m+n)*logn)

So Span $T_{infinity}$ = O(d_max*((α^dm) + (m+n)logn)

2f)

In each of the d_{max} contraction iteration – Random hook is called once, with high probability one call to Random hook processes atleast |V|/16 vertices (i.e. |V|/16 vertices are hooked) , thus at most 15|V|/16 vertices are forwarded to next iteration of contraction and retain their PhD Status.

We have $= \sqrt{\frac{15}{16}}$, thus $\alpha^2 = 15/16$

Hence n_d $(Number\ of\ vertices\ mainting\ PhD\ status\ at\ level\ d) <$ n . $(\frac{15}{16}^d)$

Thus n_d < n . (α^{2*d})

2g)

Pr [edge i is selected] = 1 / m

Pr [edge i is not selected] = 1 -1/m

Pr [edge i is not selected in m_d]= $(1 - 1/\text{m})^{m_d} = (1 - 1/\text{m})^{m.\alpha^d} = e^{-\alpha^d}$

Consider a vertex v_i with degree k such that PhD[v_i] = true

For $v_i$to lose its Phd status , all it's k edges incident on it shouldn't be selected in m_d then all these edges would become light

Pr [k edges to be not selected] = $e^{-\alpha^d k}$

Now let x be a random variable whose value is 1 if edge becomes light and 0 other wise

This E[x] = k$e^{-\alpha^d k}$

To maximize expectation we will derive w.r.t to k and equate to zero

Thus d k$e^{-\alpha^d k}$ / dk = $e^{-\alpha^d k}$ (1-$\alpha^d k$) =0

Thus k = 1/α^d

Multiplying this value with N_d – the number of PhDs at any iteration level

We get the require bound on Expected number of edges becoming light r_d < n. α^{2*d}/α^d = n. α^d

$Thus\ r_d$ < n. α^d

2h)

We have proved that $n_d < n.\alpha^{2d}$ -From Q 2 f

Hence $n_{dmax} < n.\alpha^{2dmax}$

Substituting values for dmax = [¼ $log_{1/\alpha}n$]

Thus $n_{dmax} < n.\alpha^{\frac{1}{2}log_{\frac{1}{\alpha}}n}$

$< n.\alpha^{log_{\frac{1}{\alpha}}n^{1/2}}$

$< n.n^{-\frac{1}{2}}$

$< n^{\frac{1}{2}}$

Thus number of super vertices after dmax iteration of reduction = O ($n^{1/2}$)

E' – Edges between super vertices – thus Given O ($n^{1/2}$) vertices , E' = O (n)

As for complete graph of n vertices can have at max n^2

2i)

Expected Work and Span of CC3

Work = T_1

Below steps(1-7) will be executed d_max times

1. Time needed for E_cap (line 3) = O(α^dm)
2. Form U[v] (line 4)– O(n)
3. Check each edge the sample (lines 5-9)= O(α^dm)
4. check each vertex v in V(line 10-11) = O(n)
5. V_cap construction(line 12) – O(n)
6. Random hook(line 13)-O(n)
7. V' construction (line 14) – O(n)
8. Map the solution back to the current instance(line 16)- O(n)

So complexity for these steps – O(d_max*((α^dm) + O(n)))

Steps 9-12 will be executed once

9. V'- collecting only the root vertices(line 18) – O(n)
10. E'- construction(line 19) – O(m)
11. Execute ParallelCC2- O((m+n) logm) (see below)
12. map the solution back to the current instance(line 22)- O(n)
    So complexity for these steps- O((m+n)logm)

Overall complexity = O(d_max*((α^dm) + O(n))) + O((m+n)logm)

= O(d_max*((α^dm) + O(n)) + (m+n)logm)

But, d_max = ¼ * logn/log(1/α) = O(log n)

Therefore, overall complexity = O(logn*((α^dm) + O(n)) + (m+n)logm)

Work for ParallelCC2-

Parallel CC2

7. line 1 – O(1)
8. associate the edge (u,v) with u and v (line2)- O(m)
9. Random hook(line 3)- O(n)
10. Construct V' (line 5)- O(n)
11. Construct E' (line 6)– O(m)
12. map the solution back to the current instance(line7)- O(n)

Recursion will execute steps 1-6 logn times

So complexity becomes- O((m+n)*logn)

Span Calculation

Below steps(1-7) will be executed d_max times

13. Time needed for E_cap (line 3) = $O((\alpha^{d}m)$
14. Form U[v] (line 4)– O(logn)
15. Check each edge the sample (lines 5-9)= $O(\log(m^{*}\alpha^{d}))$
16. check each vertex v in V(line 10-11) = O(logn)
17. V_cap construction(line 12) – O(n)
18. Random hook(line 13)-O(logn)
19. V' construction (line 14) – O(n)
20. Map the solution back to the current instance(line 16)- O(logn)

So complexity for these steps – $O(d\_max*(\alpha^{d}*m) + O(n)$

Steps 9-12 will be executed once

21. V'- collecting only the root vertices(line 18) – O(n)
22. E'- construction(line 19) – O(m)
23. Execute ParallelCC2- O((m+n) * logm)
24. map the solution back to the current instance(line 22)- O(n)
    So complexity for these steps- O((m+n)logm)

Overall complexity = $O(d\_max*((\alpha^{d}*m)) + O(n) + O((m+n)\log m)$
$= O(d\_max*((\alpha^{d}*m) + (m+n)\log m)$

Span for ParallelCC2-

Parallel CC2

13. line 1 – O(1)
14. associate the edge (u,v) with u and v (line2)- O(logm)
15. Random hook(line 3)- O(logn)
16. Construct V' (line 5) - O(n)
17. Construct E' (line 6)– O(m)
18. map the solution back to the current instance(line7)- O(logn)

Recursion will execute steps 1-6 logn times

So complexity becomes- O((m+n)*logn)

So Span $T_{infinity} = O(d\_max*((\alpha^{d}m) + (m+n)\log n)$

2j)

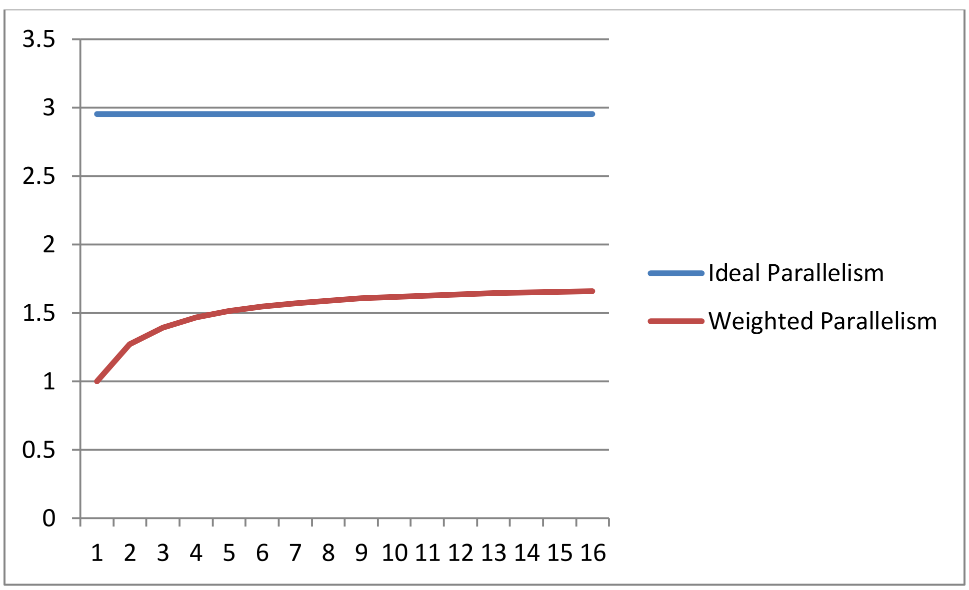
The running time for all the algorithms for all the given inputs:

Graph	CC	CC2	CC3
as-skitter	26152	13801	19536
com-amazon	1986	1015	1205
com-friendster			
com-orkut	422024	236723	272140
com-dblp	2381	1132	1466
com-lj	120193	62475	81067
ca-AstroPh	680	368	360
roadNet-CA	4906	2360	4621
roadNet-PA	2700	1256	2753
roadNet-TX	3346	1538	3275

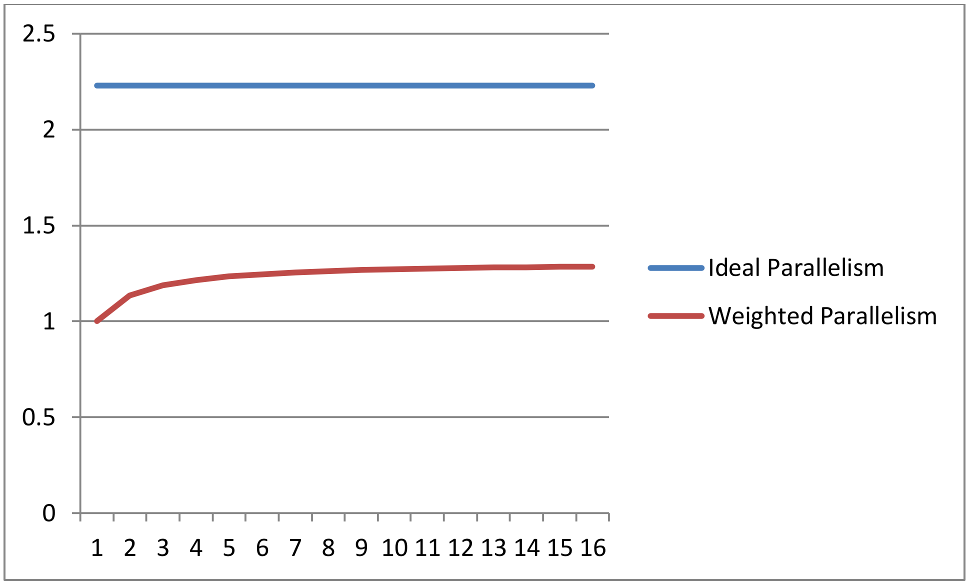
The timings reported are in milliseconds

2k)

Cilkview plot for the live journal graph using parallel_randomized_CC



Cilkview plot for the live journal graph using parallel_randomized_CC2



Cilkview plot for the live journal graph using parallel_randomized_CC3