Analyze Initialize, Span and GetMin :
Work:
T1(v) = 2T1(v/2) + O(1)
k = 1
Case 1: f(v) = O(1) = O(v^(1-1))
-> T1(v) = BigTheta(v)
Span:
T(inf)(v) = T(inf)(v/2) + O(1)
k = 0
Case 2: f(v) = O(1) = BigTheta(v^0 * (logv)^0)
-> T(inf)(v) = BigTheta(logv)

Analyze Dijkstra:
Work:
T1(v) = BigTheta(v) + (v-1) * (BigTheta(v) + O(1) + BigTheta(v))
= BigTheta(v^2)
Span:
T(inf)(v) = BigTheta(logv) + (v-1) * (BigTheta(logv) + O(1) + BigTheta(logv))
= BigTheta(vlogv)
Parallelism:
T(inf)(v) / T1(v) = BigTheta(logv)

Using adjacency list representation and binary heap, complexity of Dijkstra is O(elogv) where e = O(v^2).

If the graph is (very) sparse, e = BigTheta(v)
-> complexity of serial adjacency list Dijkstra is BigTheta(vlogv)
-> there is not much difference between serial adjacency list Dijkstra and parallel Dijkstra

If the graph is (very) dense, e = BigTheta(v^2)
-> complexity of serial adjacency list Dijkstra is BigTheta(v^2 * logv)
-> parallel Dijkstra is faster

Adjacency list Dijkstra is more difficult to parallel because the algorithm constantly has to update the binary heap that stores the distance of each remaining vertex. The binary heap can only be updated by one thread at a time, therefore reducing parallelism.

1. Consider the parallel algorithm on an idealized parallel computer (i.e. with infinitely many independent processors). Given your analysis, which algorithm would you prefer to use in dense graphs? Which would you prefer in sparse graphs? Briefly explain.

On sparse graphs I would prefer use the serialized algorithm. This is due to the fact the time saved would not be substantial enough with a sparse graph. however with infinite processors, assuming we use all of them, it would waste a lot of resources to run parallel, thus I would also do serialized with dense graphs.

2.Consider the parallel algorithm on a computer with only 2 processors. Given your analysis, which algorithm would you prefer to use in dense graphs? Which would you prefer in sparse graphs? Briefly explain.

I would use serial for the sparse graph. The dense graph I would definitely choose the parallel algorithm. This is due to the fact we have 2 processors, thus we would save a substantial amount of time when running dense graphs.

3. Why would it be less straightforward to parallelize the adjacency list version of Dijkstra's? That is, what does it do differently that would be difficult to parallelize?

Because we used a binary heap class in this homework.