5-4

(1) A BFS can do the job.

Scan all edges and store the directly adjacent planets of every planet; technically speaking, for any planet pi, find all pairs pf hops that contains pi.

Initialize a hash table $Distance$ to store the “shortest path (minimum fluid units required)” from earth to all planets. When initialize, set all value to MAX.

Initialize a set $to\_visit$.

Initialize $num\_depth = 0$

Then run BFS starting at earth:

First for all planets adjacent to earth (can directly hop from earth), set all values in the $Distance$ to be $hij$, and add all such planets to $to\_visit$, increase depth by 1

While to\_visit is not empty and and depth <= k:

Depth +=1

Initialize new set new\_to\_visit

For i in to\_visit:

find its adjacent (directly accessible) plants $adjancent\_planets$.

For j in $adjancent\_planets$:

Distance\_j = min(Distance\_i + hij, Distance j)

Append j to new\_to\_visit if j does not exist.

to\_visit = new\_to\_visit

At last, check if Distance(kernel) <= k.

Correctness:

It is correct since we tried all possible paths from earth to Kernal with in k trips.

Runtime:

For every depth, the number of edges we tried is smaller than m. Visit every edge takes O(1) time to access the hash table and do the operations. So the total algorithm runtime is bounded by O(km)

（2）We can still make the job done using adjusted version of BFS.

Scan all edges and store the directly adjacent planets of every planet; technically speaking, for any planet pi, find all pairs pf hops that contains pi.

Initialize two hash tables $Distance\_end\_with\_hyper$ and $Distance\_end\_with\_normal$ with MAX.

Initialize a set $to\_visit$.

Initialize $num\_depth = 0$

Then run BFS starting at earth:

First for all planets adjacent to earth (can directly hop from earth), set corresponding values in the $ Distance\_end\_with\_normal$ to be $hij$ and $Distance\_end\_with\_hyper$ to be 1, and add all such planets to $to\_visit$, increase depth by 1

While to\_visit is not empty and and depth <= k:

Depth +=1

Initialize new set new\_to\_visit

For i in to\_visit:

find its adjacent (directly accessible) plants $adjancent\_planets$.

For j in $adjancent\_planets$:

Distance\_end\_with\_normal\_j = min(Distance\_end\_with\_normal\_j, Distance\_end\_with\_normal\_i + hij, Distance\_end\_with\_hyper\_i + hij)

Distance\_end\_with\_hyper\_j = min(Distance\_end\_with\_hyper\_j, Distance\_end\_with\_normal\_i + 1)

Append j to new\_to\_visit if j does not exist.

to\_visit = new\_to\_visit

At last, check if min(Distance\_end\_with\_normal(kernel), Distance\_end\_with\_hyper(kernel)) < k

Correctness:

As before, the correctness is proven naturally since we have tried all possible paths from earth to Kernal with in k trips.

Runtime:

We have added a constant time operation for every path at every depth, so the total runtime is still bounded by the same complexity as 4(1) which is O(km)