一条基因序列由一个带有8个字符的字符串表示，其中每个字符都属于 `"A"`, `"C"`, `"G"`, `"T"`中的任意一个。

假设我们要调查一个基因序列的变化。**一次**基因变化意味着这个基因序列中的**一个**字符发生了变化。

例如，基因序列由`"AACCGGTT"` 变化至 `"AACCGGTA"` 即发生了一次基因变化。

与此同时，每一次基因变化的结果，都需要是一个合法的基因串，即该结果属于一个基因库。

现在给定3个参数 — start, end, bank，分别代表起始基因序列，目标基因序列及基因库，请找出能够使起始基因序列变化为目标基因序列所需的最少变化次数。如果无法实现目标变化，请返回 -1。

注意:

1. 起始基因序列默认是合法的，但是它并不一定会出现在基因库中。
2. 所有的目标基因序列必须是合法的。
3. 假定起始基因序列与目标基因序列是不一样的。

示例 1:

```
start: "AACCGGTT"
end:   "AACCGGTA"
bank: ["AACCGGTA"]

返回值: 1
```

示例 2:

```
start: "AACCGGTT"
end:   "AAACGGTA"
bank: ["AACCGGTA", "AACCGCTA", "AAACGGTA"]

返回值: 2
```

示例 3:

```
start: "AAAAACCC"
end:   "AACCCCCC"
bank: ["AAAACCCC", "AAACCCCC", "AACCCCCC"]

返回值: 3
```

分析，该题为dfs的广度遍历，从节点开始，下一层若是有值，加入队列，若等于需要的值，返回层数，若没找到，返回-1.

```
class Solution {
public:
```

```
    int minMutation(string beginWord, string endWord, vector<string>&
wordList) {
        set<string> wordSet(wordList.begin(), wordList.end());
        if (wordSet.find(beginWord) != wordSet.end())
            wordSet.erase(beginWord);

        queue<string> que;
        map<string, int> mp;
        int level = 1;
        que.push(beginWord);
        mp[beginWord] = level;
        while (que.empty() == false)
        {
            string now = que.front();
            que.pop();
            int level = mp[now];
            for (int i = 0; i < now.length(); i++)
            {
                string tmp = now;
                string change = "ACGT";
                for (char j : change)
                {
                    tmp[i] = j;
                    if (wordSet.find(tmp) != wordSet.end())
                    {
                        if (tmp == endWord)
                            return level;
                        else
                        {
                            que.push(tmp);
                            wordSet.erase(tmp);
                            mp[tmp] = level + 1;
                        }

                    }
                }
            }
        }
        return -1;
    }
};
```