最大公共子序列实验报告

学号：2111454 姓名：李潇逸

2023/4/20

1 实验源码

1.1 回溯算法解决最大公共子序列问题

```python
import string
import time

def check(a : string , b : string , indexa : int , indexb : int , result1) -> int:
    if indexa == 0 or indexb == 0: #当两序列有一个序列遍历结束时返回0
        return 0
    elif a[indexa-1] == b[indexb-1]: #当两序列此时元素相等时返回1+下一步递归
        if a[indexa-1] not in result1:
            result1 += a[indexa-1]
        return 1 + check(a,b,indexa - 1,indexb - 1,result1)
    else: #当两元素不相等时，选取两序列下一元素返回值的最大值
        return max(check(a,b,indexa - 1,indexb,result1),check(a,b,indexa,indexb - 1,result1))

def main() -> None :
    a , b = input().split()
    result1 = []
    s = ""
    ans = check(a,b,len(a),len(b),result1)
    for i in range(len(result1)):
        s += result1[i]
    print(ans)
    print(s)        You, 13小时前 • homework_code …

if __name__ == '__main__':
    start = time.time()
    main()
    end = time.time()
    print("run time is",end - start)
```

图 1.1: 回溯算法源码

1.2 动态规划算法解决最大公共子序列问题

2 实验结果

2.1 长度为 5

2.1.1 回溯算法

5 长度最大公共子序列回溯算法计算结果为 4，子序列为 bcde, 用时约为 7.58ms。

2.1.2 动态规划算法

5 长度最大公共子序列动态规划算法计算结果为 4，子序列为 bcde，用时约为 7.04ms。

```python
import time

def main() -> None:
    result1 = ""
    start = time.time()
    a , b = input().split()
    ans = [[0 for _ in range(len(a)+1)] for _ in range(len(b)+1)]
    for i in range(1,len(b)+1):
        for j in range(1,len(a)+1):
            if a[j-1] == b[i-1]:
                ans[i][j] = 1 + ans[i-1][j-1]
                if a[j-1] not in result1:
                    result1 += a[j-1]
            else:
                ans[i][j] = max(ans[i-1][j],ans[i][j-1])
    result = max(max(ans))      You, 13小时前 • homework_code
    print(result)
    print(result1)
    end = time.time()
    print("run time is",end - start)

if __name__ == '__main__':
    main()
```

图 1.2: 动态规划算法源码

```
abcde bcdef
4
bcde
run time is 7.579539775848389
```

图 2.1: 5 长度最大公共子序列回溯算法结果

2.2 长度为 20

2.2.1 回溯算法

20 长度最大公共子序列回溯算法计算结果为 16，子序列为 bcdebcdebcdebcde, 用时约为 109.62ms。

2.2.2 动态规划算法

20 长度最大公共子序列动态规划算法计算结果为 16，子序列为 bcdebcdebcdebcde，用时约为 8.01ms。

3 结论

经过运行测试可以发现，回溯算法和动态规划算法运行所得的结果相同，且两次实验均为动态规划算法运行速度远远快于回溯算法运行速度，证明了解决此类问题时动态规划算法具有更好的效率。具体原因是：动态规划算法每一次运行后都主动记录了结果，因此不需进行大量重复运算；而回溯算法则不得不进行大量的重复运算。

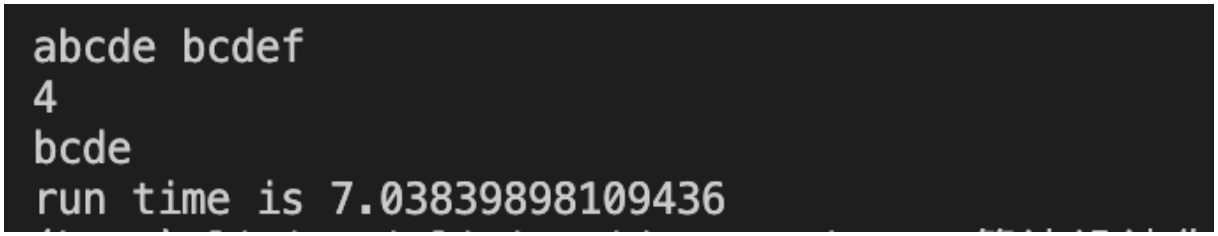

图 2.2: 5 长度最大公共子序列动态规划算法结果

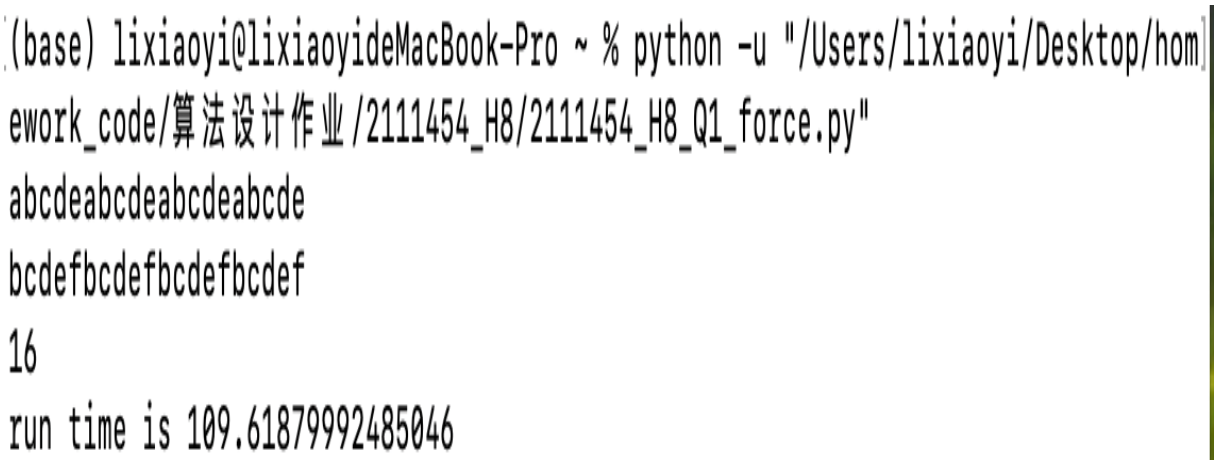

图 2.3: 20 长度最大公共子序列回溯算法结果

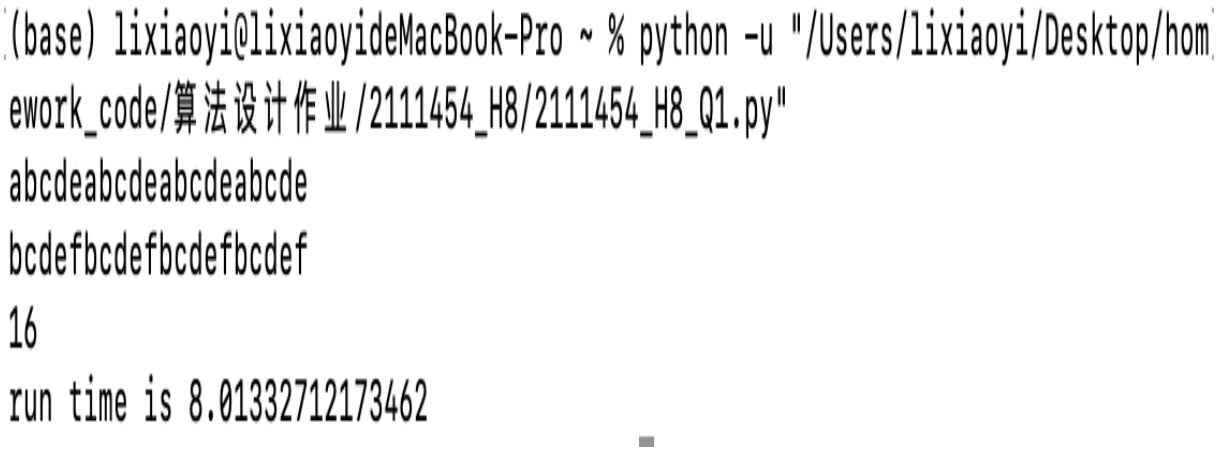

图 2.4: 20 长度最大公共子序列动态规划算法结果