# Project 1: Multiple Sequence Alignment

Name: 吴曾宇

Student ID: 519021910981

GitHub：https://github.com/DDuanCang/SJTU-AICourse-Proj1---MSA

目录

[Project 1: Multiple Sequence Alignment 1](#_Toc116074229)

[Implementation 2](#_Toc116074230)

[DP 2](#_Toc116074231)

[A\* 3](#_Toc116074232)

[GA 3](#_Toc116074233)

[Alignment and cost 5](#_Toc116074234)

[DP： 5](#_Toc116074235)

[A\*： 6](#_Toc116074236)

[GA： 8](#_Toc116074237)

[Running time 9](#_Toc116074238)

[DP： 9](#_Toc116074239)

[A\*： 9](#_Toc116074240)

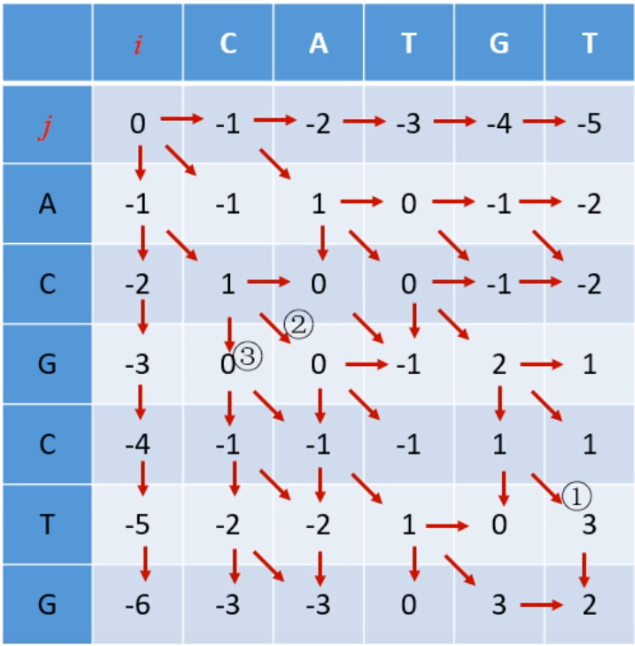
[GA： 9](#_Toc116074241)

[Time complexity 9](#_Toc116074242)

## Implementation

### DP

DP解决序列比对问题，我采用了序列比对动态规划矩阵来进行搜索。



首先生成score matrix，并填写出第一行第一列。再进行从第二行到最后一行，行中从左到右的cost value填写。填写规则为：将blank的左、左上、上方blank中的cost value加上这一次移动代表的cost，再进行cost value比对，取其中的最小cost为结果，并记录下移动方向。其中右移代表横轴进行一个GAP，下移代表纵轴进行一个GAP，斜移代表进行一次对应位置字符的比对MATCH or MIXMATCH。根据proj1任务书中给出的Cost matrix可以有，一次GAP增加3cost，一次MATCH增加0cost，一次MISMATCH增加4cost。重复此法填写score matrix直至填写完整。

Score matrix填写完毕后，从右下角向左上角进行回溯，获得least cost结果。结果中包含least cost value和对应的其中一种比对结果（score matrix中的回溯结果可以有多种，即可能有多种有least cost value的比对方法，只是这里根据proj1的要求只取其中一种为比对结果）。

重复上述流程即可完成一次query sequence与database sequence的序列比对。记录下每次的比对least cost value与比对结果，遍历比对database中的所有sequence，并进行比较，获得cost value最低的比对序列组。

多序列对比则在普通序列比对情况基础上增加一次，即对一个database中的sequence用两个query sequence同时进行比对，同样取最低cost。

### A\*

A\*的大致流程与DP相似，不过在填写score matrix中进行cost计算时增加h(n)，这里的h(n)取当前位置到最左上角的曼哈顿距离的三倍。取曼哈顿距离是因为这里的所在位置到左上角位置可以近似表示为回溯流程中当前位置到最终位置的距离；取三倍是因为曼哈顿距离的计算中取水平距离与垂直距离的和，这在表格填写过程中表示一次右移GAP或左移GAP，因此在cost计算过程中以GAP value计算。

多序列对比则在普通序列比对情况基础上增加一次，即对一个database中的sequence用两个query sequence同时进行比对，同样取最低cost。

### GA

（在经过各种测试后，由于运行时间问题，将迭代次数定为100）

此次GA解决序列比对问题我使用了python3.9版本中的scikit-opt库进行辅助。该库的文档链接为：<https://scikit-opt.github.io/scikit-opt/#/zh/README>。具体使用函数为from sko.GA import GA。该函数的参数如下：

|  |  |  |
| --- | --- | --- |
| 入参 | 默认值 | 意义 |
| func | - | 目标函数 |
| n\_dim | - | 目标函数维度 |
| size\_pop | 50 | 种群规模 |
| max\_iter | 200 | 最大迭代次数 |
| prob\_mut | 0.001 | 变异概率 |
| lb | -1 | 每个自变量的最小值 |
| ub | 1 | 每个自变量的最大值 |
| constraint\_eq | empty array | 等式约束 |
| constraint\_ueq | empty arrary | 不等式约束 |
| precision | 1e-7 | 精准度，int/float或者它们组成的列表 |

本次序列比对中，使用GA的思路为：

eg：

#   seq1: KJXXJAJKP

#   seq2: KJOBL

# =>

#   seq1: [K,J,X,X,J,A,J,K,P,-,-,-,-,-]

#   seq2: [-,-,-,-,-,-,-,-,-,K,J,O,B,L]

# =>

#   seq1: [0,0,0,0,0,0,0,0,0,1,1,1,1,1]

#   seq2: [1,1,1,1,1,1,1,1,1,0,0,0,0,0]

# gap\_num1 = len(test\_seq2)

# gap\_num2 = len(test\_seq1)

# =>

# p：[0,0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,0,0,0,0]

按照上述方法构成p为种群个体，fitness函数（即引用库中的func参）为比对的cost结果，constraint\_eq参数为保证每个种群个体中的gap数一定，lb为0（表示非gap），ub为1（表示gap），precaution为1（表示每个位置只存在gap与非gap情况）。

此结构的本质为通过GA找出两序列中能形成最低cost的gap位置。

多序列比对则是：

eg：

#   seq1: KJXXJAJKP

#   seq2: KJOBL

#   seq3: KJP

# =>

#   seq1: [K,J,X,X,J,A,J,K,P,-,-,-,-,-,-,-,-]

#   seq2: [-,-,-,-,-,-,-,-,-,K,J,O,B,L,-,-,-]

#   seq2: [-,-,-,-,-,-,-,-,-,-,-,-,-,-,K,J,P]

# =>

#   seq1: [0,0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1]

#   seq2: [1,1,1,1,1,1,1,1,1,0,0,0,0,0,1,1,1]

#   seq2: [1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,0,0]

# gap\_num1 = len(test\_seq2) + len(test\_seq3)

# gap\_num2 = len(test\_seq1) + len(test\_seq3)

# gap\_num3 = len(test\_seq1) + len(test\_seq2)

# =>

# p[0,0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,0,0,0,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,0,0]

剩余部分与普通序列比对步骤相同。

## Alignment and cost

### DP：

##################################################################

KJXXJAJKPX\_KJ\_\_JXJKPX\_K\_\_JXXJAJK\_\_\_PXKJJXJK\_P\_XKJXXJAJKPXKJXXJAJKHXK\_JXXJAJKPXK\_JXXJAJKH\_\_XKJXX

XHAPXJAJ\_XXXJAJXDJ\_AJXXXJAPXJAHXXXJAJXDJAJXXXJAJXXXJPP\_X\_\_JAJXX\_X\_\_HAPXJAJXX\_XJAJXXXJA\_JXXXJA\_\_

cost: 161.0

##################################################################

ILOTGJ\_JLABWTSTGGONXJMUTUXSJH\_KWJHC\_\_TOQHWGAGIWLZHWPK\_ZUL\_JTZWAKBWHXMIKLZJ\_GL\_\_\_X\_\_BPAHO\_\_HVOLZWOS\_JJLPO

IPOT\_WJJLAB\_\_KSGZG\_WJJ\_\_\_KSPPOPHTSJEWJHCT\_\_OOTHR\_\_\_AXBKLBHWPKZULJPZKAKVKHXUIKLZJGLXBTG\_HOHBJLZPP\_SJJPJO\_

cost: 148.0

##################################################################

IHKKKRKKKKKKXG\_WG\_KKKP\_\_KSKKKKKBKKKPKHKKXKKBSKKPKWKKLKSKRKKWXKPKKBKKKPKTSKHKKKKLA\_\_DKKYPKKKOP\_HKKBWWLPPWKK

IKBSKKKK\_\_\_\_WKKKKKKKKWWKKKKPGKKKKKXXGGKRKWWK\_WKKPKKKKKXKKKRWK\_\_\_\_MKKPKK\_W\_P\_\_\_\_KKKKKPGKKK\_\_LBKWWK\_\_\_KJ\_\_\_\_

cost: 163.0

##################################################################

MPPPJ\_PXPGPJPPPXPP\_PJPJPPPX\_PPPPSPPJJJP\_PXXPPPPPJPPPXPPXIPJMMMXPKPSVGULMHHZPAWHTHKAAHHUPAONAPJSW\_PPJG\_A

OPJPXJPPMJPPMXPPMJPPXJPPOXPPXJJP\_JXXP\_XJP\_\_POJPPMXPPOGPP\_XXP\_\_PO\_\_MPPXXPPO\_\_XPPX\_JPQ\_\_\_XPP\_\_B\_\_JPPPXPPX

cost: 205.0

##################################################################

IPPVKBKXWX\_\_\_KH\_SAP\_\_\_H\_\_VXX\_\_VOJMRAK\_KPJVLLJBWK\_OLLJKXHGXLLCPAJ\_OB\_\_K\_PGXBATGX\_\_MP\_OMCVZTAXVPAGKXGOMJQOLJGWGKX\_\_\_LQ

ITPVKWKS\_KXKXUAXPVHXVOMMKHYBPABLLOBGKOLLJ\_\_\_\_\_\_GXZGXLSO\_\_LAMOG\_KIGXBATBXMPJTCVMTAXVMPWWA\_W\_\_\_\_\_OM\_O\_\_UPH\_HZBI\_TKKXLK

cost: 182.0

##################################################################

run time: 8 seconds

##################################################################

IPZJJLMLT\_KJ\_ULOST\_K\_\_TJOGLKJOBLTX\_GK\_\_TP\_LUWWKOMOYJ\_BGALJUK\_LGLOSVHWBPGWSLUKOBSOPLOOKUKSARP\_PJ\_\_

IWTJBGTJGJTWGBJTPKHAXHAGJJSJJPPJAPJHJHJHJHJHJHJHJHJPKSTJJUWXHGPHGALKLPJTPJPGVXPLBJHHJPKWPPDJSG

IWTJBG\_\_TJGJTWGBJTPKHAXHAGJ\_J\_\_\_\_XJJKPJTPJHJHJHJHJHJHJHJHJHKUTJJUWXHGHHGA\_L\_KLPJTPJPGVX\_\_\_\_PLBJHH

cost: [273.0, 54.0]

##################################################################

run time: 3 seconds

### A\*：

###############################################################################

\_\_\_\_\_\_KJ\_XXJAJKPXKJ\_J\_\_XJKPX\_\_KJXXJAJKPXKJ\_J\_\_XJ\_KPXKJ\_XXJAJKPXKJXXJAJKHXKJXXJAJKPXKJXXJAJKHXKJXX

XHAPXJAJXXXJAJ\_\_XDJAJXXXJAPXJAHXXXJAJ\_\_XDJAJXXXJAJXXXJPPXJAJXXXHAPXJAJ\_\_X\_\_XXJAJ\_\_X\_\_XXJAJ\_XXXJ\_A

cost: 165.0

###############################################################################

ILOTGJJLABWTSTG\_GONXJMUTUXSJHKWJHCTOQHWGAGIWLZHWPKZULJTZWAKBWHXMIKLZJGLXBPAHOHVOLZWOSJJLPO

IPOTWJJLABKSGZGWJJKSPPOPHTSJ\_EWJHCTOOTHRAXBKLBHWPKZULJPZKAKVKHXUIKLZJGLXBTGHOHBJLZPPSJJPJO

cost: 158.0

###############################################################################

I\_\_\_HKKKRKKKKKKXGWGKKKPKSKKKKKBKKKPKHKKXKKBSKKPKWKKLKSKRKKWXKPKKBKKKPKTSKHKKKKLADKKYPKKKOPHKKBWWLPPWKK\_

IKBSKKKKWKKKKKKKKWWKKK\_KPGKKKK\_KXXGGKRKWWK\_WKKPK\_KK\_K\_KXKK\_\_KRWK\_MKKPK\_\_KWPKKK\_\_\_KK\_PGKK\_\_\_KLBKW\_\_\_WKKJ

cost: 175.0

###############################################################################

MPPPJPX\_PGPJPPPXPPPJPJPPPXPPPPSPPJJJPPXXP\_PPPPJPPPXPPXIPJMMMXPKPSVGULMHHZPAWHTHKAAHHUPAONAPJSWPPJGA

\_\_OPJPXJPPMJPPMX\_PP\_MJ\_PPXJPPOXPPXJJPJXXPXJPPOJPPMXPPOGP\_\_PXXP\_P\_\_\_\_OM\_\_\_PPXXPPOXPPXJPQXPPBJPPPXPPX

cost: 211.0

###############################################################################

IPPVKBKXWXKHSAPHVXXVOJMRAKKPJVLLJBWKOLLJKXHGXLLCPAJOBKPGXBATGXMPOMCVZTAXV\_P\_AGKXGOMJQOLJGWGKXLQ

ITPVKWKSKXKXUAXPVHXVOMMKHYBPABLLOBGKOLLJGXZGXLSOLAMOGKIGXBATBXMPJTCVMTAXVMPWWAWOMOUPHHZBITKKXLK

cost: 186.0

###############################################################################

run time: 11 seconds

###############################################################################

IPZJJLMLTKJULOST\_KTJOGLKJOBLTXGKTP\_LUWWKOMOYJ\_BGALJUKLGLOSVHWBPG\_WSLUKOBSOPLOOKUKSARPPJ

IWTJBGTJGJTWGBJTPKHAXHAGJJSJJPPJAPJHJHJHJHJHJHJHJHJPKSTJJUWXHGPHGALKLPJTPJPGVXPLBJHHJPKWPPDJSG

IWTJBGTJGJTWGBJTPKHAXHAGJJXJJKPJTPJHJHJHJHJHJHJHJHJHK\_\_UTJJUWXHGHHGALKLPJTP\_JPGVXPLBJHH

cost: [285.0, 54.0]

###############################################################################

run time: 4 seconds

### GA：

########################################################

KJX-X-JAJKP--X-KJ--J-XJ--K--PXK-JXX-JA-JK-PXKJ-JXJ-K-PX--K-JX-X-JA---JKP-X------K--JX-

IP-JTUMAOULBGAI--JH-U-GBSOWBWLKK---B--G--KP--GT-GWC-IO-BGXAJ-LGT-WCBTG--LWTKKKYGWPOJ-L

cost: 265

########################################################

ILOTGJJLA-BWT-ST-GGONX---JMUTU---XS-J-H--K-WJH-C-T--OQHWGA--GIWLZH--W-P--K--ZULJTZ--W

-M-L-T-W-LO--HSBA-IL-UZMSGG-STWWLO-G-CAGL-L-SHKBWALZ-UM--ATHBP-Z-XOP-HUABKPS--TR--POP

cost: 271

########################################################

IH--KKKRKKK-------KKK----XGWG-KK-K--P--KSKKKKK--BK---KKP-KHKKXKKBSK-KP-K-WKK-LK-SKRKKWX---K----PK-

I-PJ-J--HM-JJPYJJH-HGJJHPPJ--JJ-S-GJPHJJ-S---ANH--JJPKPJH---H-J--JKJ-PK-K-H-JJ-P----KP-JJHKXJHH--J

cost: 292

########################################################

-M--PPPJ---PXP-G---PJ-PPP-XPP-PJP-JPPP-XP-PPPSP-P--J---JJ-PP-XXP-PP-PPJP-PP--XP-P-XI-P--JM--M---

CKTWLPGJTJJG-LG-JLS--P--PA---XAJPJ-P--J-PJP--J-P-JP-JPJP-JP-J-K-JP-J-XS-T-LXH--J-JOSAKPBHVBAGCPJ

cost: 276

########################################################

----I-P-P--V-KBKXWXKHS-AP--HV--X--XVO-J-M-RAKKP-J---VLL---J-B-WKOL--L-J--K--X-H-GXL

ITOJ-L-LLULST--KS-P---P-MOG-KJLWXOGOOKWL-L-SK--P-KWSOB-VOQGRBS---PPP-BXUA-LLPKPO--M

cost: 256

########################################################

########################################################

run time: 1116 seconds

## Running time

### DP：

2：8 seconds

3：3 seconds

### A\*：

2：11 seconds

3：4 seconds

### GA：

2：1116 seconds

3：由于我使用的方法对计算量要求过高，因此在迭代次数与种群规模较小时计算不出结果（代码是可以运行的能获得部分解，但仍存在问题，且由于时间问题，没有时间进行大量计算得出结果了）

## Time complexity

DP：O(n^2)

A\*：O(n^2)

GA：与迭代次数(max\_iter)有关