Project Report 3

To Buy or Not to Buy

Date: 2023-4-13

1 Chapter 1 : Problem Introduction

1.1 Problem

- **Problem description**: Given a number of beads in the store, we need to select some of them to form the one we want. Beads are represented as strings, including [0-9], [a-z], and [A-Z].
- **Requirements**: The selected string has the least number of beads (characters) left after forming the desired string
- **input&ouput**: Each input file contains one test case. Each case first gives in a line the string that Eva wants. Then a positive integer N ($\leq$100) is given in the next line, followed by N lines of strings that belong to the shop. All the strings contain no more than 1000 beads； For each test case, print your answer in one line. If the answer is `Yes`, then also output the least number of extra beads Eva has to buy; or if the answer is `No`, then also output the number of beads missing from all the strings. There must be exactly 1 space between the answer and the number.

1.2 Background of the algorithms

- **Depth First Search (DFS)** : An algorithm used to traverse or search a tree or graph. Traverse the nodes of the tree along the depth of the tree, searching the branches of the tree as deep as possible. When all the edges of node v have been explored or the node does not meet the conditions during the search, the search will backtrack to the start node of the edge where node v is found. The whole process repeats until all nodes are accessed.
- **Backtracking (exploration and backtracking)** is an optimal search method, also known as the heuristic method, according to the optimal conditions to search forward to achieve the goal. However, when the exploration reaches a certain step and the original choice is not optimal or fails to reach the goal, it will go back one step and choose again. This technology is called backtracking method, and the point in a certain state that meets the backtracking conditions is called "backtracking point".
- **Pruning optimization**

    1. Optimize search order: In some problems, it is possible to simplify the unsolved subproblems by analyzing the subproblem branches by first solving the relatively simple subproblems. This can be achieved by optimizing the search order.
    2. Feasibility pruning: If the search direction obviously does not contain the target state, stop the search in time and turn to the branch that may contain the target state
    3. Optimal pruning: Update the current optimal state/optimal solution after each search, and judge whether the current solution is worse than the last one before each search? If so, stop the search and move on to other search branches

2 Chapter 2 : Algorithm Specification

2.1 Depth First Search

Main idea:

1. Select the initial state and start the search from a bead string
2. Traversal of the legal state generated from the initial state or the current state, that is, recursive search for the next bead string
3. Check whether the new state is the target state (whether the required string of beads is gathered). If yes, return; otherwise, continue traversing and repeat steps 2-3

Pseudo-code：

```
Procedure dfs(index):
    if(index==final) then
        return;//到达最深的顶点
    if(Satisfy the condition) then
        Record the result//满足条件，记录结果
    dfs(index+1)//搜索下一个结点
    Backtrack();//回溯
    dfs(index+1); //搜索下一个结点
}
```

2.2 Backtracking

Main idea:

1. If the search reaches the last bead string, return to the previous level
2. Restore the changed data

2.3 Pruning optimization

Main idea:

1. If the current number of extra beads is greater than the previous result, return

```
int ans_temp=0;//每次搜索新结点时加一
int ans_size=0xffffff;//符合额外字符最小条件时赋值（第一次直接赋值）
if(ans_temp>ans_size){
        //如果当前搜索到的字符串数量已经大于目前的答案，直接返回
        return ;
    }
```

2. If the current string is not sufficient, return

```
for(int i=0;i<256;i++){//遍历每个可能的字符
        //如果待选字符串里的字符加上目前选择的该字符小于需要的字符，直接返回
        if(remain[i]+current[i]<goal[i]){
            return;
        }
    }
```

3. If the number of extra beads is already 0, return

```
if(ans_size==0){//如果恰好不需要额外的珠子，直接返回
        return;
    }
```

4. Sort beads according to their matching degree to destination beads before searching

```
int value[string_number];//记录待选字符串与目标字符串的重叠个数
void mysort(){//按重叠个数从大到小排序
    for(int i=0;i<string_number-1;i++){
        for(int j=0;j<string_number-i-1;j++){
            sort(strings) by value[j];
        }
    }
}
```

3 Chapter 3 : Testing Results

(You can find all test results in **mytest.txt**)

- Case①（Sample Input 1）:

```
RYg5
8
gY5Ybf
8R5
12346789
gRg8h
5Y37
pRgYgbR52
8Y
8g
```

Result:

```
Yes 3

the average time cost is:0.006554

the max time cost is:0.001045

the min time cost is:0.001045
请按任意键继续. . .
```

- Case②（Sample Input 2）:

```
YrRR8RRrY
3
ppRGrrYB225
8ppGrrB25
Zd6KrY
```

Result:

```
No 3

the average time cost is:0.000512

the max time cost is:0.000512

the min time cost is:0.000512
请按任意键继续. . .
```

- Case③（10 random selected bead strings）:

1 I8m4N1Z84385JaTybXEU7X73445PD1q5rcmR3k02T47brF2Bn5mFr95IY1LmlhPi500W4HVfnoimAF82404H7u4L5nELpfGAg4I7U6Q3JCs3i4LMIoqK56SItA5Y2m7190eE5cvu8B34A7vPJswc5a87JiL5kbye9W16e388uF3Pp5BZBK6kwCv5Ca244Qryaos57R3u6GP9lMO69tnvB4H1a9iH2563fyr4tRm1iz4V8naZkE1TrA051N11EbHB29i4H0WpG4YblMfv

2 10

3 676k0WmCiJ1Zy6DaUqf7gKB4L3L41gOu2ei1UGf0Qkwft3

4 SR9E7L5GHac884Fz3POvpDnU07BEok10AP94B18QzmH6ZSLbf3Y9Uc3c6P4yPB6Gyd7MSdY99PSP0MQ78ygxdy30RbF2qbEJoF7xmD33P7ecJpC68Ho5YZmBXgo3b3q9MKxAVjr4pWz8JL91M5iY1PQPaG75E53D331B3QJXchq87O3ThIs6tLd6pb51iH9TKl2Uf26l6ncxdJ55xa2HHg1hrc5S3xkvT1E20G2g3CVSjTy3OHUACYNh09G20bTBcMhwd9BLol1d4kIz2yB60Iiy4Fbzpp8Z4v2eo1O90575lO8SKe475g27uRhEiEYbA2wWL8Im7g1i8Qg1d0zm3o2895xL1W2uOwZ53d5AMqh0b7YupOecpWVNl196gkhY8ri0sv5kjF92jNtHlay1CMpzmZ8tLnk7NtI992PM0W9pngaTWBUxo4A9tr4IXNeYi7nwxIQ278ZdTvOs0J84NvrT03V7W28k6mBe80S68r4nz5rdfP2DAnwhu58v3YW94yWGMKp6H68Lm01v2cdr24fq6MO8eEuQ5P70MaIW9ROz304ULu8w7v4fqhxTvnRzeEPA5O4xgwxe6M3UULRPK7pUuKnRqdDo1UeBjut7c1pt51xOHXgDxmo5qdh2H1h5bLv9QUFTUyGSDJmN61wa6J030xGYc424sq13659XTbt3jyVz5yKOM2rJQBQy8UA38zAeqI6u7fZkXdK2HCMPUtcC6p01dvXN4848B1qH16Md57xPYUa8nZbhHP45Srm8743hA6ifmrW2YE17djru99m

5 t6Xce33uv39NKQBn45PPpt2J4UWb0lrC517k39Furk28Jze0aN9qNsC0W25h

6 6FQ7202m5eZBvs3hu4pSN3ipnhN8ebC0qPHIgF3Lw2HvTBa5n04k7TUD71l2KYuHTNFQE7WW3Du38OJFMp116Wu6f3B8qaVj71cbAC3B35gfOwQUU5A7eIk2p95Y04IQS6VwGh6QW8PD0K1AJd6Aqpl74vS8ux45f

7 8s39qC0iseYU3Gz7d52ybTd8a4k9ZLCUE6U153bgY3TY45Ogwqkx1BKVnfPMSBxuZoY5VNdgAhXtP2Ps7cSkj47uYW3E92F3C9AW88w22NRE6V7zUnC6zY550FDA0453kJGqM2WRdh6Vt678uzz0IwZZLDc3Wbtn18drP4u076kdD83JIXdI7Tp59w79SUlCnoWbc195i41X54X5S48qC709c8gMKHXAHr9EJiTcq5A98fQ1km35nY93KW53gi21cTcU2niFb9O5MlTnbX34F71gJZ47BY19G3Q262Z13WZhnIW7biCt4391jhw7a0521BBAwB8hX6v2s1JY4RG4MR0gn2ew9ycen3n4WfchzCJDN6fQAlhMB6I7g567ckSouE45JnuK522yB47dhh7Vl9aTGX53yvoAiZ4qH18xOBG0KOXx59K7U9w1ZWm6k8Y6QMjVmcG6bDJK82BwX3TgP6w504T3wDTP0UCH97h34c06bKWg1M627QDpA6s2iknS619j5aY4s6MKv5zaYDt5cIgOLPPRXCeNT1Hf2934OrF72RIIRmrAcnZz3zvbeY48Of9n80I2oZlD1QRx4D9S8X5kP69bX8NKJKhVh4SQQddaEIPeZ2RtQcl32dx30L5Ui57Ed81L0FCCGX3djEX5FL63Q14Piljo8IwBqI0G9EgHrDxigYIzWpZJFQB64C19gZpuxfEnS179v1qSahf8Vw1xQ48FShGp62cygja4H86lYBm09fCeM267xo3C855O26rLdOT3BpHZ8QzGNuAtu618Fl6pF6UoF02xP06bOo12l7q0ilB4tfMbX5UXmUj7CcOz95Mr8e918Ewj79Z5IlJ0t6ap53w97roW59rnbIz6j7X7UaHHEkPfd5

8 5k6NaLeez69l2Q5OgN9B7swx84j3FV1T32CQLVoY3g8d3X6O3Zt5sh4n7hSkxLm33R9hfNNA1i0tZBGSTJ2EK518ZPXAt9NFBn08oyp65OY462TaZUScr7l31Ec7SdURz3j372vcAYY4PDWd8Y65x46BnL7qGwbPjHNd7O8K4fW5UUCAghOW14xR3wqaBxy6Lv3IUKca24nKZ1xaI24a4bz033tP041uwZzfb0K45da64qVB2MeRSA0URSTzo3hGU9D0Qu3652nDt37nul69l2wAbp6r3HWBYnijvQd74Ge340iYNXyN4hmzu416b9073GpGig4KhsnyU043m73cfd1D289mD7mM124tFROjtO8zT9n

9 wDWKQfl2zS72N3JzX44wb7LpiB4Oh9kHngmvZ54N6ud4g0n900VrIy875fWtcl67lsX57dD2iS7cbPiys32rF85FLXyFD5qw3t602Xom8TX95O3t4JfTfaY0NhSjfj0wNuT0gQ5PiGF8pL30WL66Gi5Kv2r306LHtRtjBzeZjIBFQ7xPLaU8p61NGNBf9373E1OvoL8J3wWIPbg5Lns1dYr3h0k33y0WfV69Sy0uGJn63rlL0f6qt19ESqJW9H8be9Xb3k30rj5227X3YjA6C4NAL9t5y4149h7u60jxBQC41SIxhmD6Z5GSZtn8No3ipCE9dwYKFVQiobMARUqRSupeM230JwZbN2dwOb1t0RY7B4W1pg7MEm4C9kH5s8T35VYBy6al70SGrJdU520y00d7nrtPW5bMfLqTYdQbhEXth1B5dY01Yi1Uz2OeUwE2U89OEeEt2wwM59kJ07Ams5h72bv8tj3XTS3A0Qod5qDy8jhbjc418dB1XIe3jMECSQ6X1xQ4Mb2q9Lt3E36CFy4IBGTMOHU3vMaPEmIlt08q5e234uFEC3wuigptRDmVtHXg9s58XTOWj1sii26U8JlCksVJo16TP6y7wq83p0wby2U1OWNgbh6pa970Qds1Rv5wW9J27285W5204200Pq6EkI8g0k6fTEDc0ux4Vgr3kvtt6SS1XNX3BP2AwvflOJva8W3KG0lUYBuPVBQAhlDweC7968s6dZtusf9f1Gcm8pbd2oYr4GnP3p785B17mblm9dRnAVaX31w53JZ3c9i38no9CqVeB9pPKfI48G518g6Ntc3kr4fXv5E0lw6k38c3D7kf5VpAVz46Z2YgR3JTdl6hMWeM39pKAs115210fMM9xOZMmbhETi09q67SN8y20hFE94mpvd2e1al36R0Iy0KDr3fUl9OL2aIkaxASGnHxzJCb0B8jkPLFc00au98dk3k6VCBO9zCpczOMo4Gud

10 1kGYKi09o8lXpD7j3gYTuZZYUlbFP53G7vhjqAea7Xsw7L1E4bX2cE4h13MYB71ttEHQA6r3JyPr2LBJzYCm329lu82qnqZ542uPMk5V7b8gy1XbO78zJhhhsY7znyD991X2p9nJCvPB3jCE14dk16HM5KPvn636QaT7vFMmLq1B03QPeyC6s8Uxk17xu8H89Ncyn9eB15FAThQFj175o9Zm2JakLkg3j2cHt2eH725X5PRwQqM9l95Dl6eibuX6r24T5QE3537l1tnmi6aVMpl3R94FrU84lSg09vUB67868e8Xpfs2u49KD3Xu6O29rt7TxecY8F3wN3o08n1uvGlXD0hOjpl7U0gof48m5I52jM6erNs17yQD7WIfCPzXd68dYT2F2o1PlE5L25w2B0P0Ns48YdPtsEsi2zr2W5itN

```
3abl7Q5w033512c3jKOrHHZJ638M6OTD9X9fk2z6kJ9A82s2oBrUJhfN5BbOOyqPVP18DKRZfeOkbd5826AJr5I67cHSMRMh8EDLt6xiflnS81zy3rD2s64euv7m03r7em2Ml1JYBwW7iAYDjiTs1jRf0L38NE7sjLlOzwif4A59pH1Bb85XaY7LXm6v3ieeir2IXxbpe1jc307I90vncRznLEmYlBgTJdBsAfmfZnz9t28H3K2nE5fTxC7lsR5f9R3C1V7C4wn1AbMfoK8soOEVBl5rd04CLP79S834Skz8inVb1I5tS1uKwTm953h2nep8
3133tdvsOfAapx8xLvOL2AN9JAseH1VedHKGGDi34ztuU9Np7I7RIBAvRhOQiDj0n207HNN7CNZQiGP57BVht6k79TG3rP561yHwHxg2EB4jTtU35x95lIg9V5s61Lx1BHrd67ACQ49JjWa2xgsSn0n5z8Qai5uUjQPU84p44dDfL12ntAC6M62r4UC9S9iiroQE0GGT4eeO0d50EPyKKGSd9Xo93j08vx4B04ahPrpqkwfNIYJzoLMF8QvxMII6K7Rd2d703aYDW50CaCzz5s8uL7GGOm8SWZi9IhJLHR56m393bp64W3JY72hk47m1Nw2cY47Op85L089gdoWLSg34TlS5OKk9wUpucnziZqFEx82u2oyfiO0A5S6C26aJMbvsBU1UL9an6Y2dPh52Ea9Yz5tcP77eN2tgi3VK3dXYV1E3NzFYlZ1yAL49ls5GXw8W7cx0ZpfpKfTP
```

Result:

```
Yes 479

the average time cost is:0.002016

the max time cost is:0.002016

the min time cost is:0.002016
请按任意键继续. . .
```

- Case④（30 random selected bead strings）:

```
UOUKRYv10hnz73MMs67Rxy1dv930UoPl2u3B
30
EenIe8FFoVZC04GUlDMvaS8hcY4YaPFe1D0quRMi5MrP5ZKkZr699zt60xo4YkTqwa7z2UMIGgMf9lE8288nsxkcEQzP4VEZQH87Qsslv2KXQ6B96BHW76NGGFUF5raMsYu6mhM1nZGJ64N87nL1iu027Pn7YB6304Z545JZ324f8MId5kG90r8WMo8xR7Q7fXJdHw5ww255I7kDy49Waf8ZPKM6tqEMhjji0m24baLw9Tbp68gyUyihqpy74Lb08WyQvf8Zxrk Jt4PP2ZY66FeKsRU0Yae83mtsr8LtYv26M1jZ2SG2bM7kIeUJYTb777J1EdZm1c7YOXPB72ET79rtozL3P0IF5uH869ngiDhzDcSu0E8nY5i1IoU31UUcYXAY4TBEmf3vB6flL1JR4721B5J38sCBh9AqXZVO1GL8Fs0wqP1OXVzWyBohbasTnp0lsbeW18oER15mE2ln5D1m3R8mThY0ipOdcS1DS8W1K5yUR1EBP6zf0FK5cW5ML7Kx0FcUO2NWCKAC59rtW38Omyqq6ZSOexfdT0g7v1n5tf6yaulbc6exkyGLCAr7hs2Rp3ZlcO647Z448ll8J873EX993nu1TL96xW9eXJqUs8fXNk6Xc123apmBtPYE58fD1A0226OC6Im6QJ80x9k5b2PJ85kBfk281MaqocO6YA3vb9b7Z3NjbQ8573Y9H0yXo7d3645R4wqlB
v9Ac4w76Vpop84J322js92yf1ZdYaEGW30NHH5sXnPc64
8xO91ioHOfeIz6KNxnLBYHNh014a805066AeL0foisps8Bn3h7YzD0J8fj9i141pZqfa1hmGT5x5ZK0YB2T3NX45kjehhDMRrh26qR8oxYXdZ5u8mwU3406KMa7DsnKEcZk1Jue8354zl7Sh1BlE1JsxOiPF58euppNR2467pp7xZvkSID7WS52h1H6U3Vmq5g3lOn2mzfrsA9858GDl0K27Ro8uppHB4WX95CtAQ930k83Va859A2974krVkkN7SXpb3f1Q302V7B850ABAcDcFX7l6jowGs6GgbqebeY2HHi75aEE4UlBU6fS21RiD36Uv2yelWnx0sj87w1nUDWibKvBZ0x1npgl00c39JL8040Y550iDDvuQQ40vmyx10BxSZa9rxO8qOqMaQ3JB4JkpkAZfUg36r8W76dvmtm3myWebJ41Ws6SHcfrAZY7V98Kp4XxwCmpE6ooFDm395G9a9Y9Gd76JMLi1ac61NAPz1rxutLbK141nE83VI580Me0WB5hK85dAO4iC97KIdi6391aK1aD7Evd4QLWuTFGP7X525qMs35596Hg0F9j67eEl69PeAAV5t444LSbosqlIwxNzKfq3P6fxV19ka485N0K1sf0Aj5p7xgzFQgslSgoi43Q2EDFdTPGRp2eDAeqd96890h
Z8S1xF458ILo4Afg8d5oa4OU1L2A6sP2o76XPAzo2hq3Q1J22Ib4Nu8es8NGf2Itr6RaG27iUnb6qsAGz75qEjkmZxcpU9KZgNiWg2IuC75lIK65Z4yY813oh3k154C8GmO87JqCW5Dy2elHj98al5fsD0Pi881Hf50h0H40Pf4wu0W02hdodI2yu3Z9sZx8Meprq0L9ss15o8d2avuu78R8kSTz27PEo6i2X9dIj5v8x77sPRW46nzDX9loK9L0t176093rAyn4ltdTY7D1SEcldSiXq6ND96CC9cyaETNpXhUhg36oBVm7B1pS3aqo7kS7o1G5gtMvv5j
```

ff7MN3k564975reho20ilLK5K0l2newW60HAuB6OrGOP1atTrJ4YLjy83T2nVlHJtcN463AgArfBewEzEibz4g1U4z1OMzEUEv4NDDg1qERlL9iDQ0Z3j2d0EzZv1j682vU2hB5Wh903a9M04P01ct229Dv3f72X64bXAGgN1aG4vDmNIRBlWrPOrQ5Q21606nn6accZV8l95I4OeVPbxFj3OU0YyZJ8OUU8Pj1nOAPQ5UfF2kgWE98c6duvm6oU9YU7uFFpT26LxNTnrESPwfC9dT1VQ32p62eWWExFked586f7y9k6msAN86I6VIi61420no1MBNdlUOMfzcP1Ld1MdkK5O2734Abmk8G1SjSPLLm514r9eW2oV6I0v2Ht8w7bl22AgL09JJLAfBQjzn2ppyf4oMDP35oO4z4rqK8kcQzly7xz7nQH78w56WT1ii2Z2C19S0QJ4yKiBGl5tT9wPdydQ4Oron5x05ym8j1scl0XJC0uVZ932Yv66zc1vBBZ6e2k53Qj61PnqRc376bsol4sQC1gdDjtiqaPMC77itgZ936CPEtF31tp572DHvLXL2UL30K072CHSf1aEDlVZ3465hQ48J0Q2898zR67mGhhmqdb9K898IcLx7k8u6cRl00PIY9fjTF6JqMbbgJ7mjMwY9Aap4xMIMXJaQ8cBW1zz8Hpe8CkYH2mRHSMTt558F0qm3umYiquScfMms7cgwh2jDoyapkK6oCULR63ax8zsQyh6Q9DFf16rYf668QcWk9gRl9T9bRpLJm9sK1aSiC5
g32d0G1UiL1d
0aEHZg9Cw97Dd54nm5A8Yp62Wm11Xivsy4M8SR9jp5mzS9cO93750b2owR36lwSZBdQlKv4
57120T0wp0zzG2I3JeqM16tX10Gnu1r4ZvZo5xkLYAe4F3jJ65g405r0o71cAh45OH5UyzKsns93xFn6bh3to4sgw5lwo00wS3skXpI1f8kjNJpqDuTKLizOmGCyFK1t6pAFa9c3LV
BRFw2i3V8SN5c83t9nL1dc7EADvZFVj3C1MvCB5Vu5jBULX6EUTw3rC9zaFzRw488uJwvSrXc9LV7Qg52y8P9Yp9gpckYY057bj6JUf6s01G3Tf9or8B0WM4ukfe05cJ9QB3oNGyBPd4i90J3lXxmtWY8jvp0Nmdit33oDr5qLPpA9N79U051NK8663T5l5b273H0fe3m6Q75LHj8w0nsxwldkO4G95aCx32kgR4T8x8XubWtqx06M1l543qK0QYmVm7HmPavgPY36TKXe3N77B64Y9zG5ozHt5WP
161q9d6SWG06Vi6a406T4AhSNUd7iq9j41JS66e37xrv6udnf1G2pkp9w2IrI0B83hb6k33TGCTMChn77yZebo67QP2u4atEdPEaajIR4hly8KeepqFjI38Hc43T7Xy6BFyOb86Y546hS37UYl96jJsZXrx0NjHr0bDrd938eT6nimXo1gOscaNM909n11kXTk5CsF5kWoNDF4coM6MhD8LIigmQncZtzD9r7g9RCfQ2aM3F7r7de6O7FeAj9M36n5GyGoE71JnknkmWvpmA73SGrIc8xHOVPBgCz04k0Yb9683gwmK5kN8Gg4W8lY7Dku49IfXZjG386ReW5lvH
gpXAhuYHZJgx4risMh6s7u2M3eqDJTvB6w8Y8c6RnR0118vREANqD6592idp8mv76v4EG48C37IP4HI571rS6WotSbSv9z5A12d25Lge54qg26KYkREYK9jB9lwcH3288MijO322q8dY5GsHu7O1ojWHpyx1AXdx0k7S7wW4B70fJoCJ1xtuIS0M3h947zM5J2EjyQFVakJafUGoXD5XBE0iU7a5E9TguB8Y2Hm8QIudNrKN35EpUfE10W865WhD5zMN2fNP247xCSxeti949RWNfGZAVJ1ie6Ato37862xdspG7I5PsE97e6ENW5e5IWfx5b7Ys8iP902oJ5Zs6wMyaWJ9yL720o3F4MZm6S47FLl3xmA25qcz
j4aupEFfQ3rKWulB3848fmFT6597Ck8h8gaC5F59fj4iexng96YVnzg0xRkd0CM3VUM5b0zA0I7M286td5055wc0WdA4E6R20I58ltp1P2q0f65Pr2cayu7ph0pBIauxw19WZ821R6t9dPd0X9I7kq4E2gUtCEfhGUkf5498y4EsHBwp6MzYCykU3Tsa89ia03v52L8fquTN4nr161kp00X11a94qZYtAQHl6WF5MhOkmgkYA6X1qROHJ10Y6wRwH2iKqePFPqUWd8n6890FFge5hFn23LSb6czWrHBg9a9Y41NA036262HWnJgy7927D4Qz1HF5L9Wu0WNKuixWX27UyYUYUerHWeJ6ycHk39Q8vB10e7UlS5adq9I8fhjBmtzPa986js9nrVdm0OAto0jUqjK8xVh1S4FWYHDbEJPYDrsK19bf2nZeRBw7wWZ565UlzQkO7FSfoDF6LMT8xVQ99frzXGrnWMCWK9r7U4B8eb7lD95s82zhcxYJhJ9JROH7l5zodC9S7Z44I0E6827T9Gp48032AG15aM77UEiWIFsZK6M6Wwt4e2Ummm14h017CS1eD33149zmPdD64Hi8X82P1363ycI9Fs1X03LS1jyS1MYFfpxR4vMhoCMqSgAnsemRWZokzK6794ByZa02p
2oA80cp3n3RxDM8QPWH90pBiShj0qc4ujZ7zWkyWc431Cgf9UEiX4fFHeBf39E7verORY5GoEv74FTtZoDOAFRzFF37C8hBW650v1yCTiWFMxnRm3ucPALtM95464N5bcf7jK4gX1Kcxt6PeD1xs78JV44j4i5MDyd4bT10702pUN56F53cvsm0WYngkOdmNmWtOe97IEycz11R0KEGR7N2URw7svmzxjRCxq54U5uWCUsqriKBtj1g19GLl41wtr6OAD6P9f7gqJaXkZ757t2zJJ2Vl92gU8F0
9mDCgp0AJ0GekDovGiX52w1t5ivYIay2jIWYwVw44ZD9HD1HRnWP50gb8hN7S4B631KdDY90v9u66Nrv3wq2MT7GP49izgS32O7IwXn2Lg93L7Mxso48ep2X0fE5oAY7ec937w86TR4449JZLsXz7ci4P1uanRrj7dM0VN7Yilzp2h16J7R1GbtlEJVByeCh5rZ8IRoeZ65tqY0l0QKLK1D9JGIQ2xk9919S010cXaYsZeGk2cNmTq73O4YrjH5165K8M4c74a5ANw0fU8m03gS34HPb28rNG3S657JV9VskWAhT94mYU2e999TpQAah694ftRq1RBz
EWTZCd9O2pKGMO2cw8wwaLz9Pva8DPjsB42mpE1Qjq8so446s2ke921oB6KJsH0Sm2kv7R2d27122HM9MrnQnbn4Qr2y9n65b6w0NU34WFdJL7dUMpmlyd3dunlYfc5649YAKwe9142H7M9Fo8tV9OPiZ1Pj0pzbDqwwFA4ej00uc7CG7XYvj0Q4632VEM1Dzta0HOo8A5wZz4s08qO943fG0Zd8Yh0v1DrwuJq8W5gNs694d8Ol65k45EbW
0TM1x
UtVHIaHsDUATR4eEaY2oDOjD8177N9l83T6g240h3wT75wp443x1vC96AC2fId0k748N75ricErW9bPwvAPlRvi3itXzyQhpkIPHSd97Y1aIW8dkxUye9iOz9PPGG9ufI55d5D4t3m4dBT6Db954hJdzBy07J4cOuH969MQDD696q8syMZ22sP2F6Cyw4Wnh3JyPiH5zFO2lxeXI5H2q1KE3BLuxkAtzEBzzm99c9H17lAYNZMYTqUnQ3l15JOV246ac53cJ0lxyO07bv206tezrAg4bG4IKElLJ1aJw08JOeSL30hmbjXl

20 6Lq1Vn4oHG83rf2HnGBWiarat1pZ7bgzZew202u6y394ynM1SbEfsJ5jMRI678Oyx66z4OG6d72f4w4xWBS8Iwhr3
DKDb2wgH9yxg5xEWX3AV1vo1911Uf2yXWV6u90AHlZ06RI4594993L6dqIT6Lm74u1tK6c359bNI57FooQw5Q3EpU
04o77yK0j4nPNDB7P31i8s6I0sKi60zm3cWTo8410b76GuPDRA60RsRb402A2KBa4Uln7OjT8DLqZLl9JQoE2Z8Yy
g95deLSt1jEqBhOc8iAmp2h7I27tT7aVa8FLCD485hjPuTV4D5NWA8V5UdFhMc3Y1YQHS144EFKkY7syyGt44nm72
Yegl97B4TeszKMWnnZpEn7wNBw2I494A7U67q3ihecqR3jwL6lb0gGYcfMi7cOSKQ4M8NOOwBBeN87aOj3HkjN6ha
cLDd142K5pCB47Aw2d370jKH5v1k2Ls7HS0Q5xtuTP22nC26d56f3WQYl77FWhS17P1rVZ3Knjlrui22WYJYT1B4u
g5B7F9WUM90tojwI3P5kb2A44FB8Cr3Jei74lZmFuSnj3C30b36fb96n2XbDXZ87yEPiff8To6KY26278J25N34Kj
8SREH27pNrX4486oAx8Xp75p795Er4IVQy9tLlKl2Y5ZqmNmn3CyS4D51wKw62z52EoDcoPGK52bKG4D7E77Nki31
GAREhXkB5FmLqq9l21e9Y9W8k7C6E0213h4z1NMfl220X5SJlbq5079fQuCnaGD8eC8KSzXMu005N5MtLi5ORL8Jt
7r06CYkIm3sz79E3Jpt2BQNhtOC1ZiIhBed9oyZXX5ZFHXuag
21 77u1IsvJoqP0vce6JHnd81T1aLqpkyV6Cwj7g5g5n8B3Isja38f7vBPB5B86TFf0j1wN9r2Z5q4pfgm03THC14m3Q
JKy5jkC2XmCQu8lL31Fbht9Cj60dELSUUYEmKyRCJox3284t827F8N8WNZx4YqEwjF4rRHOehYNB8GhpZo4vvpMvh
YJSqoczM7jYm4723IY5M7MYUoer45guAu6HKH192oGtn7eAf1Fvbxin2Wpq4nP8sr99BHOjHL17LvkKKD0X2HfgqA
61qy5W9Nwkct4d9JI3hwudNltwp8pP2VG22RFZM5E8926k914vzaQ08EJMA9pv22Wda4JCYmqsO7TCYV
22 7e3x1duLvB57QvSPFlzqE6smwoqq9SRM8ZK9URb0Oi371SnymbMYCrn4pCIqHt8oY1KXl1gnb82432fKr1N0h9cms
7uw7qFEx8x3UGtd9U0Dj1600juo8rouRfeS2X2UTG0AFKC3hV9f68cU4s0q4A14huh0h38QtaC19C3bQju934kueH
exmw5ay7nm45viGNB9r05r3cceP95tQM7fnX564bLI5BF0WynwRCMwn7XUPV3a43p9hUKWiksUpNjlAnzuLPu3l4r
A9ea4H92KdnS5HP8X07a3FC09IYWqP93dKnU7Y94sYK5tJ45W7x239IEUrbHnnAs5YfiX97mZQFGR31xZ9Lv4y914
9EVRCJ2oWk3oohd87K4r1W97I72q680R3Glx6ENz4FADS8n60dFJtlOcS4gfa590M4wJY0si0055Y7575zR1CUwiw
B3K1wyZZ1SpPacdbLeS0gn70Q1N1068LU2J17hsS4T25wHi062447rRLLi7pR3L3B1
23 pCPNS0E0Pu2Y5YYpVHAl0K7hw9EX8PSLHzMTdzQgZ4I3vkZ21593W9ZmuD86LmZxrwhQ236KZ7sB6u6yYdyZL8jJ3
qz5mVy26t47FDS20p8dZ3vLwfj121oLIpL55J3RTSGiauoxlzd4f6q85GPnMXO0ju67955k8KjwU2NUEd0pCBc4QZ
Hc46BoW7nriJpBRY7S8lw7w4T7s6K940p3A2mA3bIIq2D9x2M8gCvc9fOih4zfuT54KMu3bLL25QTBdfs04T5T7Y1
2jQ3nqBifeFZc7BH1X7ud9853JvFTzTrKGiOGENx1d11qlm1dO5JU1lONIU9pmhj65CX3AR2Vc6x2q8zWhq81mm90
fF7I41WNC7S75Fhb9N2j9v9608yw25B6E95419s6hrQr8khIDtkF4kkgOgJB7Bu3o7A14B9bhpAYjftO187D5m34C
6hknZ47u0Fk8d2ctGkytEn590Ita08sUo2K6Rk7m3E1dcdS95wrarPapOwv26BX4bQS4yZV8Yq7npne1f36EzmCH7
v08YfsB2nuiRSct225zVo0yNE4qj91o83bvoH1s1UzlSTskcW1q2Z690PXJL65Pq1wXJ7h7135JhIhlAwn64dZyiS
L6vNR2Imp6G66sHo00OvP2q0BtC9785KjXeRq4PA73zKkdwo23Ep0Q95QGsC132j44L4YuzbTyKvg15E3aOHmfBkR
673sH4FkH6fFM9Af020Nltk30qNcUMUp8wE1J5iNb7Ls2dw05swuSokHeP0c827WPVY725OYxjeQerh2dwJ1ErNbk
00l11pg840jr
24 oA68D9h10fJ5LJxwq7Bl8g9ex5vR2wF3g0SpQ84Zk3sdgEv4Dx889sT6S5NcyavtPt84hbFAvTZp2c1DLkoMoiG60
ki62Yxn4hVyx2WdgORxnRRp8f52E487r538049jO5nyp38cW1ucI17tKz7x7bE0039jYmCo97I2VAYZ0MVB8EDY0z
7btlFtsGS9G7415HOYto8IW937dR30eFox6TLjDV62M0IJa30LWjSj8N2umrkNHA5Il612370xv9xS7DPOPv6Sl3I
61W56K628pKHIF6v16klYPzDB67mQFNb9SVsUzlpjK17S2kt49kw7iC8058a3nrc84x2NYs33nIpn2Feg6CYk1MD4
16Ie8QOwYy954m6BbIhVF8JvClIkf2RE16zqkrDDxITLmpDUi5dgkoJ8V4TVgF8orQw3PAa7707oB7TA6dY3gt8K8
K8U2z9Q2W59536Pq0kA2lI2D10zb262u6B93g6xOC49n5Ts8Yan4UK96VP4642M1yYA679abOwb457s64jI4Kc18x
UC7gLi7285cLNb8mHLD1uM900Tskey
25 49MVT5bc9dHMrX2fcILuKEVhQ0SXY9rn4eyUEtcfHeF2g3Z2goTGw22gz24Du4IV975Eu0pNs1H5aFW9GBKvJIYws
87Jt8Jt5zV1bhYbH3mHmE3KO
26 jc863ePP3Ht8Eh3p4r2uo0sgXK381yuh1mDp424t2h07NHVuJLO9Khpo4868xVsAkC7t1PJ7s1A4UI91W4K6C8
27 UMcqq33nii8Wxe72WY5K0630S3RPokw0qKLPxhMX3UV9zjG5qvWAFlkXx0q12EXoGh81EJI9GI0dW99Mh5MFos1qk
3mPvT223072C5Fg4CFMY8ejlp1NCFO4EzLHZ0RM1TptqJ8tsdT6fmhWs7P2QzRUost0Yxt3I4N7kqIGaQ9jdlWS3j
DsUKv86I5rG4UCCqf6cJYTQn3KoF2MFGD9LwF971gh5sITtos0Y445ds0CIEIo19U453nw6F08ZzOGDiu0T9i9U2z
r5141acPduJxLVE1855FYNgNwi12tlUCK4C3xB48g9ee4WzS63I77JvSnM29tk4D9S1im9Hz4PHkn5SQOOOORiH8k
rP42G2i8v13r4HhOXEBD92DZqnXOn71BBDhR9vkM3T8V794O1v91lx5RR4jGwu32dUlm2QYqdBXOuEG7Pl85pRPC1
d6hVvH8R60XgP66KL7W9HJXdhuucu82yB0q636cGBIxKEzuE2dVv8Lu0V4CUkw1al2Ma51iFJMd7U0ZrTCHd982FU
n7VJZh2VF3Au4eMil3M83U4sSS7Mx1Wh133Witux9bVCJ1lO4XWe81QNjjv4gPj4A6KJBrTOUzjbI0bc9K3UbIL54
Yw66NlpxeG376t69r6UKVHf0HdL18bbnr9EMD5O8SKx3tYL1W9PWotp7K7qZyRlNBoi4AF8WCmX9EU7W7052QN23V
89t4MM27v72u2g6lwX5H3KoCG7QT4J378sIaIK6lvM3RscGR7NQFZgdpWhV80b3NJe875j

```
28 2qk85h8914uHRp8NMDNbqR0r36iq8cdUQ15LoK1eAChPdZIbpNrc7UBfCiJ58zNrUbk94imW8kyIsm0OV8dcohd9U
   p55ueMDoZ1u8D8K8ky74voZVkDa851b6RJC0WWMpbgeU3X6DZbE4ccR8B1h9E4muxoK1O40697syVjEv7vpc2S80Q
   OWI26ng2Cw5A38o923MBhJhL7rI7g2F29dj77EbYVPD6vF194Lcksf48aB5vOIIb2U2Ug72RTxi5wX1VFe8bElOWs
   4S55Lk15eS45SN232859jmaU54DC29ZbqFo1d5tfH2tIX2gU11186FM0jp12iDj0LogL6tjB0H4fI4r2cd2wK35Ns
   68qXfn15MlJ7k71HpHa2C71Z9KEVC6XmDF9s0kg7qU9j3lVOSiV965oO2Hu94a8q88wFI2fEA2zOR1eZC17p6qS33
   9EHiFv74Hb6I4Wuy7ZkRbrq2oCd5076k1TeLw
29 5HLTc3dXM47NzHQ2oW9708nxw84UY92Mt8e0848e87xn93Sc6pL6A3fACX4E1ezUefD6Jc2Jnq9pJy3gLMWe26p9q
   12w10Sq295xg9749Rrm297f8NrbgrGD1CFrWx3bKORXl2x0gr6h4mYch1MV6mV3nEbE4JhgKiB7PR74utiCLDcD8T
   9rtHW5S4vDOo8P8CulFBreKNHb3Dle43zAMxtLx8hEl2qQ9uutx1Mp8LB0cnq0OyjAlhBozmdCvkiuEf1snZAgZOy
   Kon5fKE3d4NLM2JE4o0L6v6zd5397HpgaoX7105To0OLeCoRAXzs8f3a22094aoawC25yT0MbGel7AD82Z15WUC88
   v9GbIxCRt35QZ46NmRIlNJ99jf4bAYS7Y2oPzHPQ84bPxX0R8EL251R0M09s0bgpg1yL5V6JNuWc2zz7pyIy6P7D1
   RVvJ52fnjaxfvVuWPvPXsSBu0j775f6R9D5CIP4l51K1tEX0bG1DA24301l5J3aUJFYHKy5wfF0S873v9h1n1gVs4
   7aG4249pzM8fx68ZKnpVrv7C43sW7Mu03pOLbO4EwAl6b3DMM
30 AE4nQ33La9eclM4c0gb9snA2Z3xX9K454XLL1SbmF39uVR8d03vlhz0P2WPeD70U07ns83lA0Dr60tcKd37q62q06
   dJ84uL77RzHEa8kqQuDf02p81Kv9K2U8hDCP4ZA8K7rNFekXelJR7W4askZg2sbH5FdPRXjnOxwEw5UFoM79Ypw19
   W9UddT9WRZ7D28Cc8JL9435bD1T7QOid7yEr7dONh2cToaa0R2aeqVALr1d5jsV7xkqd80uWn39r1J7Yc33wzJOQ1
   1492YIl0M0BzWI0gv8Sh06QZrYSPI5K8h8jIpRRGn7jXGj5N0AZn7087Hz2N4okEZWJiYCEM515yI1f2S3X0wxO7L
   J2T1Kb5813IZ3nrChZqY1D5iXJsrT1aj4bZONIPK1Fs4xo4Hyt4zA2V7JkAAh3F0dmr5zxLJC1HGkFXKS87gU2tXK
   X5vusJ5E7R33M6K50s8a9C0BuL4FlxQo6vrYTH9rcNHScK
31 rFc9z67xk763tY34WRteO0A2PI35kbexEJmFgh27UCZ8XIK2V6yx8vNie11DiE3ImzN6CR4OZgoViUmm5D7DfQ3L1
   63Y7791P2qb8K1e8CTJ5WmY38CR7mW7Y1moq1Z4nu2so4F2h21o4JEPxa0L164ARiK3BCTyq5eB0CQ8CqVA59E1r0
   8yUrRwhS62vSb6CWg15F4im7V8zA3muOS0jRhRy70drqearU0FQT5Fqi3XvOXhr9KhikYLsouXf2r2y54XXD6CsjB
   2jxAJIpPHnbVDLo3RHJuj4htT0bbl0q8o6b0h6mJC64VjTeWlD8VRZx6s2J1G5c1wmrJht840Q6EO8TfJ0z7ir017
   YCk8Qe2Jzz3BH5yQWrSNpMWER43cz85Scb0QbM6Fa7r70DNS8RKsY78OTo7VPliJqg72cu2p8ATCQTpvPSTh018SC
   463UWsxtV3I4b21pJIxFR1n701Az5036uUHhewQ75c9G8GVTGzEkY4I6l6mXzC8c0DsY13fdDl2S1VApcuEfgYmU6
   9SyFDsUb8413q3O92l8n5l4JRDuB3L3ZLgcyMKRl6DPTx4nhey1iauJNlrUc6n5ds357r5T3EOeN0sTS3w4iWa3bq
   0I2tdHI822cmN5Wz6pw941TlmJcH5f2t42CVXt8KHL5MCw56ZDzhTSKtgiljVyDhf70p2Y9PSjQS7ff23byzp8V01
   eYntu678fKTi4h5C66uY5xDf07b8bb7b4A5O0n0fCevjnUuHC66KF86
32 9INFhc1a8C1ocDWkIlx1183j8Y8R3w5yEKVUytq9VcDfsz8TwZ510Pi4vs2P2oH1Im3Ood85013FqoDYVv5li09Q2
   8gBIqX307H0opX5QDH6JqQLlMXsjj7y5We9HQ48dg2K4HPi19XGaj4sF1H29M55501tE9821sHY53fN2hNWY6egK4
   IaGFiecM0DYIWiLEH3Po48k0G1u01MChG7p2bom07t9b5hJAd2wzkrLT6343ooH6UfLsOQ4zw7bE5LNI23FHdnf8j
   1ZEhYU7Dc4WIyD5YQ7qUrT77kdmsu3PyFz04VMMZm98757FqxUg83jXGcS4OP990vALE856Gg5wePevkOc3EN3rIO
   MjZ5nt15JBo1NXH5AtV8N1CDf8T1gC5Eoy76Zhm1tJ2Sp9Q7S9Khr9h1ZOnh40h038b4jPkd0RP4s9jBXPg3gbKC3
   o7DkZZxVvdrl3DuXPtT6Ws2k6q0ClHH4maI8XymI
```

Result:

```
Yes 246

the average time cost is:0.003104

the max time cost is:0.003104

the min time cost is:0.003104
请按任意键继续. . .
```

- Case⑤（max selected bead strings）:

The data are available in **mytest.txt**

Result:

```
Yes 302

the average time cost is:0.344128

the max time cost is:0.344128

the min time cost is:0.344128
请按任意键继续. . .
```

- Case⑥(pta(Top Level) Practice: 1004 To Buy or Not to Buy - Hard Version)

提交时间	状态 ⓘ	分数	题目	编译器	内存	用时	用户
2023/04/17 14:31:06	答案正确	35	1004	C++ (g++)	1080 KB	6 ms	

测试点	结果	分数	耗时	内存
0	答案正确	17	4 ms	840 KB
1	答案正确	5	6 ms	828 KB
2	答案正确	5	4 ms	828 KB
3	答案正确	3	6 ms	1080 KB
4	答案正确	1	4 ms	828 KB
5	答案正确	1	4 ms	828 KB
6	答案正确	1	6 ms	964 KB
7	答案正确	1	5 ms	896 KB
8	答案正确	1	4 ms	984 KB

4 Chapter4: Analysis and Comments

4.1 Time and space complexity

- When doing DFS, without pruning, the recursive program goes through all possible cases in turn.Since each bead string has both select and non-select cases, the **worst time complexity** is **O(2^n)**. After pruning, the program runs much faster, but there is no definite time complexity because it runs differently in different situations
- The maximum memory used by the program is the string of beads stored in the store , so the **space complexity** is **O(N^2)** (If you consider the number of strings is fixed,then **O(N)**)

  For different pruning methods, time tests were conducted respectively, as shown in the table below

Number of tests/string length	No pruning	Use all four methods for pruning
100/5	the average time cost is:0.00077308 the max time cost is:0.002028 the min time cost is:0 请按任意键继续. . .	the average time cost is:0.00079282 the max time cost is:0.002474 the min time cost is:0 请按任意键继续. . .
100/10	the average time cost is:0.00725512 the max time cost is:0.011911 the min time cost is:0.003 请按任意键继续. . .	the average time cost is:0.0027004 the max time cost is:0.007148 the min time cost is:0.000257 请按任意键继续. . .
100/15	the average time cost is:0.187899 the max time cost is:0.298982 the min time cost is:0.055806 请按任意键继续. . .	the average time cost is:0.0106022 the max time cost is:0.054693 the min time cost is:0.001 请按任意键继续. . .
20/20	the average time cost is:5.83334 the max time cost is:9.79242 the min time cost is:1.71409 请按任意键继续. . .	the average time cost is:0.0150784 the max time cost is:0.095551 the min time cost is:0.002081 请按任意键继续. . .
10/25	the average time cost is:187.497 the max time cost is:337.113 the min time cost is:88.6376 请按任意键继续. . .	the average time cost is:0.0342851 the max time cost is:0.082588 the min time cost is:0.002398 请按任意键继续. . .
5/30	too much time	the average time cost is:0.0254736 the max time cost is:0.072136 the min time cost is:0.004349 请按任意键继续. . .
100/40	too much time	the average time cost is:1.34259 the max time cost is:26.1553 the min time cost is:0.005149 请按任意键继续. . .
100/50	too much time	the average time cost is:5.95286 the max time cost is:90.5292 the min time cost is:0.004678 请按任意键继续. . .

Number of tests/string length	No pruning	Use all four methods for pruning
50/65	too much time	the average time cost is:7.18935 the max time cost is:22.0812 the min time cost is:0.019564 Press any key to continue . . .
20/70	too much time	the average time cost is:9.05195 the max time cost is:32.3284 the min time cost is:0.036251 Press any key to continue . . .

- In order to get the influence of different pruning methods on dfs search, I deleted each pruning for testing and obtained the following data:
- "**No Optimize search order**" indicates that the search is not sorted before searching

  "**No Feasibility pruning**" indicates that whether the remaining bead string can meet the demand is not determined during the search

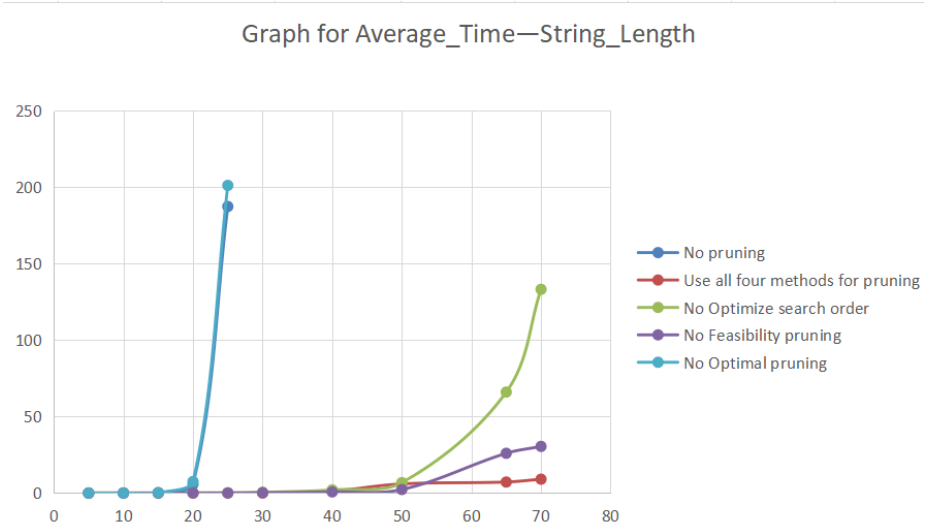
  "**No Optimal pruning**" indicates that the current additional bead is greater than the minimum additional bead

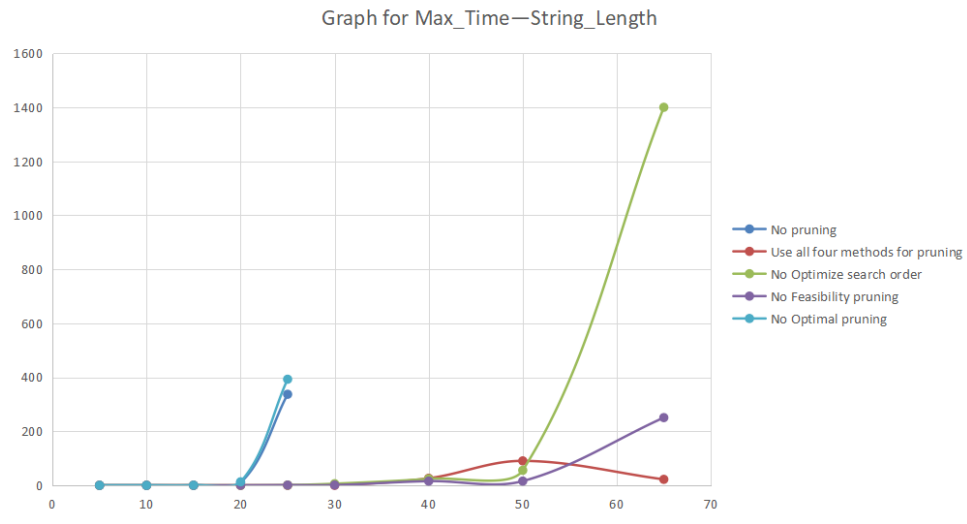
Number of tests/string length	No Optimize search order	No Feasibility pruning	No Optimal pruning
100/5	the average time cost is:0.00079476 the max time cost is:0.002165 the min time cost is:0 请按任意键继续. . .	the average time cost is:0.00086615 the max time cost is:0.002035 the min time cost is:0 请按任意键继续. . .	the average time cost is:0.00100641 the max time cost is:0.002525 the min time cost is:0 请按任意键继续. . .
100/10	the average time cost is:0.003095 the max time cost is:0.008225 the min time cost is:0.000506 请按任意键继续. . .	the average time cost is:0.00284846 the max time cost is:0.006467 the min time cost is:0.001 请按任意键继续. . .	the average time cost is:0.00894017 the max time cost is:0.016249 the min time cost is:0.001625 请按任意键继续. . .
100/15	the average time cost is:0.0130043 the max time cost is:0.073253 the min time cost is:0.002047 请按任意键继续. . .	the average time cost is:0.00977122 the max time cost is:0.045318 the min time cost is:0.001867 请按任意键继续. . .	the average time cost is:0.246157 the max time cost is:0.415655 the min time cost is:0.112925 请按任意键继续. . .
100/20	the average time cost is:0.0518128 the max time cost is:0.406487 the min time cost is:0.002534 请按任意键继续. . .	the average time cost is:0.0276303 the max time cost is:0.183753 the min time cost is:0.001506 请按任意键继续. . .	the average time cost is:7.53217 the max time cost is:13.0275 the min time cost is:3.16543 请按任意键继续. . .
100/25	the average time cost is:0.114054 the max time cost is:0.959977 the min time cost is:0.003017 请按任意键继续. . .	the average time cost is:0.0898312 the max time cost is:0.779891 the min time cost is:0.003054 请按任意键继续. . .	the average time cost is:201.272 the max time cost is:392.838 the min time cost is:94.2611
100/30	the average time cost is:0.482971 the max time cost is:6.05499 the min time cost is:0.003038 请按任意键继续. . .	the average time cost is:0.212277 the max time cost is:1.10531 the min time cost is:0.003005 请按任意键继续. . .	too much time
100/40	the average time cost is:2.04746 the max time cost is:24.0795 the min time cost is:0.002503 请按任意键继续. . .	the average time cost is:0.716971 the max time cost is:15.5268 the min time cost is:0.005802 请按任意键继续. . .	too much time
50/50	the average time cost is:6.79235 the max time cost is:55.6954 the min time cost is:0.013039 请按任意键继续. . .	the average time cost is:2.28428 the max time cost is:15.6752 the min time cost is:0.009076 请按任意键继续. . .	too much time
50/65	the average time cost is:66.0409 the max time cost is:1399.71 the min time cost is:0.01095 请按任意键继续. . .	the average time cost is:25.9867 the max time cost is:250.773 the min time cost is:0.007768 请按任意键继续. . .	too much time
10/70	the average time cost is:133.256 the max time cost is:860.632 the min time cost is:0.080508 请按任意键继续. . .	the average time cost is:30.4984 the max time cost is:128.835 the min time cost is:0.021505 请按任意键继续. . .	too much time

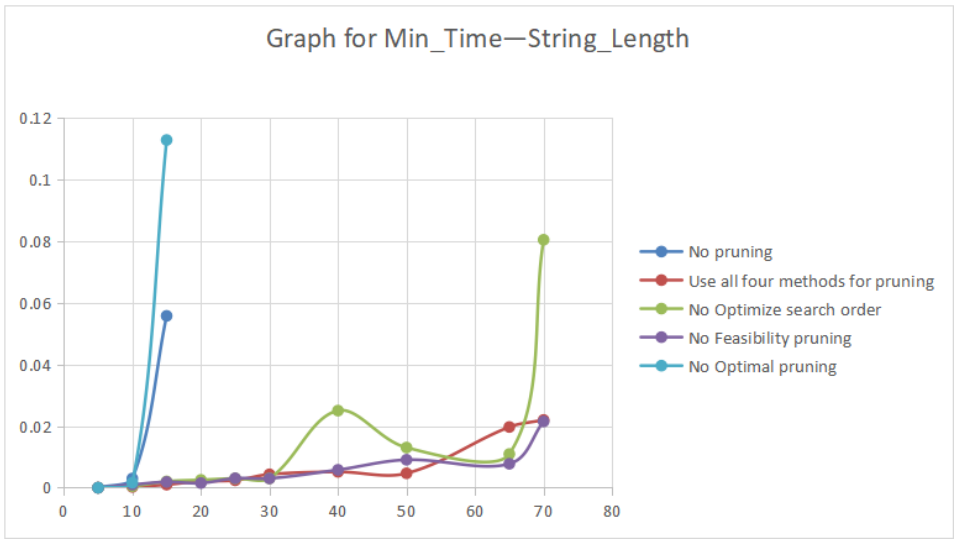
- From the above test data table can be obtained under different pruning time and input string length function image

  note: Abscissa -> Number of strings

  Ordinate -> Time(S)


Graph for Average_Time—String_Length
250
200
150
100
50
0
0
10
20
30
40
50
60
70
80
No pruning
Use all four methods for pruning
No Optimize search order
No Feasibility pruning
No Optimal pruning


Graph for Max_Time—String_Length
1600
1400
1200
1000
800
600
400
200
0
0
10
20
30
40
50
60
70
No pruning
Use all four methods for pruning
No Optimize search order
No Feasibility pruning
No Optimal pruning


Graph for Min_Time—String_Length
0.12
0.1
0.08
0.06
0.04
0.02
0
0
10
20
30
40
50
60
70
80
No pruning
Use all four methods for pruning
No Optimize search order
No Feasibility pruning
No Optimal pruning

4.2 Comments and improvements

4.2.1 Comments

1. In all cases, the overall running time increases exponentially. The rate of rise varies due to different pruning operations
2. According to the results of unpruned and complete pruning, the effect of pruning operation is obvious. The situation of unpruned exploded exponentially after more than 30 strings were input, and the time increased significantly so that the results could not be obtained. In the case of complete pruning, it doesn't happen until after 70
3. In all cases, dfs exhibit instability, where maximum and minimum times vary widely depending on the input
4. In the four pruning operations, pruning for exactly matching conditions only applies to special inputs (such as several test points on pta)
5. Of the four pruning operations, **Optimal pruning** has the largest (or basic) impact, and the time without this pruning is almost similar to or greater than without pruning (other pruning has some additional operations)
6. In the four pruning operations, **Optimize search order** has a stable effect, and the lack of pruning will slow the search to a certain extent
7. Among the four types of pruning, **No Feasibility pruning** seems to be an unnecessary operation, which is faster when the pruning is not used (due to the operation during pruning)

4.2.2 improvements

1. You can reduce the number of searches by removing any input that does not contain the target string
2. Using divide-and-conquer method may directly reduce the time complexity
3. You may get better results if you give enough time to test.Some inputs take a lot of time when testing in large quantities, and the average run time should decrease significantly if the number of tests is fairly large

Appendix: Source Code (in C++)

project3_No_pruning

```cpp
#include<iostream>
#include<vector>
#include <time.h>
#include <fstream>
using namespace std;
int goal[256];//Target bead (string)
string select_input[10001];//Beads from the shop
string goal_string;//Target bead (string)
int select_length;//The length of the target bead
int ans_size=0xfffff;//The final number of extra beads needed
int ans_temp=0;
int judge=1;
int finish=0;
int loss_judge=1;
```

```
int loss_num=0;//The number of missing beads
vector<int>ans_num;
void dfs(int index){
    if(index==select_length+1){
        //reache the last string of beads. return
        if(finish==0&&loss_judge==1){
            loss_judge=0;
            for(int i=0;i<256;i++){
                if(goal[i]>0){
                    loss_num+=goal[i];
                }
            }
        }
        return;
    }
    judge=1;
    for(int i=0;i<256;i++){//Determine whether the target bead string has been assembled
        if(goal[i]>0){
            judge=0;
        }
    }
    if(judge==1&&ans_temp<ans_size){
        //If it's already done, and the number of beads needded is less than last time,
        //Reset ans_size
        ans_size=ans_temp;
        finish=1;
    }
    for(int i=0;i<select_input[index].length();i++){
        //Traverse the current bead string
        //and reduce the corresponding number of the target bead string
        int temp=select_input[index][i];
        if(goal[temp]<=0){
            ans_temp++;
        }
        goal[temp]--;
    }
    ans_num.push_back(index);
    dfs(index+1);//Search for the next bead string
    for(int i=0;i<select_input[index].length();i++){
        //Traverse the current bead string
        //and add the corresponding number of the target bead string
        int temp=select_input[index][i];
        if(goal[temp]<0){
            ans_temp--;
        }
        goal[temp]++;
    }
    ans_num.pop_back();
    dfs(index+1);//Search for the next bead string
}
int main(){
    clock_t start,finish;
```

```
    start=clock();
    cin>>goal_string>>select_length;
    for(int i=0;i<goal_string.length();i++){
        goal[goal_string[i]]++;//input the target bead string
    }

    for(int i=0;i<select_length;i++){
        cin>>select_input[i];//input the bead strings in the shop
    }
    dfs(0);//Conduct the search
    if(ans_size<0xfffff){
        //If we can get the beads we need
        cout<<"Yes"<<" ";
        cout<<ans_size<<endl;
    }
    else{
        //If we can't get the beads we need
        cout<<"No"<<" ";
        cout<<loss_num<<endl;
    }
    finish=clock();
    //cout<<endl<<"the time cost is:" << double(finish - start) / CLOCKS_PER_SEC<<endl;
    return 0;
}
```

project3_with_pruning

```
#include<iostream>
#include<vector>
#include <time.h>
#include<algorithm>
using namespace std;
int goal[256];//Target bead (string)
int current_bread[256];//The number of characters in the current string
int remain_bread[101][256];//Number of remaining characters in the string
int sell_bread[101][256];//Beads for sale
int value[101];//Record the number of beads that are the same as the destination bead string
string select_input[10001];//Beads from the shop
string goal_string;//Target bead (string)
int select_length;//The length of the target bead
int ans_size=0xfffff;//The final number of extra beads needed
int ans_temp=0;
int judge=1;
int finish=0;
int loss_judge=1;
int loss_num=0;//The number of missing beads
vector<int>ans_num;
void dfs(int index){
    if(ans_temp>ans_size){
```

```
        //If the number of beads currently purchased is already greater
        //than the number of additional beads purchased last time, return directly
        return ;
    }
    if(ans_size==0){//If there is no extra bead string,retrun
        return;
    }
    if(index==select_length+1){
        if(finish==0&&loss_judge==1){
            //When the last string of beads is first found, count the number of
missing beads
                loss_judge=0;
                finish=1;
                //Only calculate the first time, and then do not calculate again
                for(int i=0;i<256;i++){
                    if(goal[i]>0){
                    loss_num+=goal[i];
                    }
                }
            }
        return;//reache the last string of beads. return
    }
    for(int i=0;i<256;i++){
        //If you can't get the beads you need, just retrun
        if(goal[i]>remain_bread[index][i]+current_bread[i]&&finish==1){
            return;
        }
    }
    judge=1;
    for(int i=0;i<256;i++){//Determine whether the target bead string has been
assembled
        if(goal[i]>0){
            judge=0;
        }
    }
    if(judge==1&&ans_temp<ans_size){
        //If it's already done, and the number of beads neededd is less than last
time,
        //Reset ans_size
        ans_size=ans_temp;
        finish=1;
    }
    for(int i=0;i<select_input[index].length();i++){
        //Traverse the current bead string
        //and reduce the corresponding number of the target bead string
        int temp=select_input[index][i];
        if(goal[temp]<=0){
            ans_temp++;
        }
        current_bread[temp]++;
        goal[temp]--;
    }
```

```
    ans_num.push_back(index);
    dfs(index+1);//Search for the next bead string
    for(int i=0;i<select_input[index].length();i++){
        //Traverse the current bead string
        //and add the corresponding number of the target bead string
        int temp=select_input[index][i];
        if(goal[temp]<0){
            ans_temp--;
        }
        current_bread[temp]--;
        goal[temp]++;
    }
    ans_num.pop_back();
    dfs(index+1);//Search for the next bead string
}
void initial(){//Initialize all the data used
    for(int i=0;i<256;i++){
        goal[i]=0;
        current_bread[i]=0;
    }
    for(int i=0;i<101;i++){
        for(int j=0;j<256;j++){
            remain_bread[i][j]=0;
            sell_bread[i][j]=0;
        }
    }
        ans_size=0xfffff;
        ans_temp=0;
        judge=1;
        finish=0;
        loss_judge=1;
        loss_num=0;
}
void mysort(){//Sort the bead string by value
    for(int i=0;i<select_length-1;i++){
        for(int j=0;j<select_length-i-1;j++){
            if(value[j]<=value[j+1]){
                int temp=value[j];
                value[j]=value[j+1];
                value[j+1]=temp;
                string temp2=select_input[j];
                select_input[j]=select_input[j+1];
                select_input[j+1]=temp2;
            }
        }
    }
}
int main(){
    clock_t start,finish;
    start=clock();
    cin>>goal_string>>select_length;
    for(int i=0;i<goal_string.length();i++){
```

```
        goal[goal_string[i]]++;//input the target bead string
    }
    for(int i=0;i<select_length;i++){
        cin>>select_input[i];//input the bead strings in the shop
    }
    for(int i=0;i<select_length;i++){
        for(int j=0;j<select_input[i].size();j++){
            if(goal[select_input[i][j]]){
            //if the character of the bead string is needed
                value[i]++;
            }
        }
    }
    mysort();//Sort the bead strings by value
    for(int i=0;i<select_length;i++){
        for(int j=0;j<select_input[i].size();j++){
            //input each character of the optional bead string
            sell_bread[i][select_input[i][j]]++;
        }
    }
    for (int i=select_length-1;i>=0;i--) {
        for (int j=0;j<256;j++)
        //input the number of characters for the remaining bead string
        remain_bread[i][j]=remain_bread[i+1][j]+sell_bread[i][j];
    }
    dfs(0);//Conduct the search
    if(ans_size<0xfffff){
        //If we can get the beads we need
        cout<<"Yes"<<" ";
        cout<<ans_size<<endl;
    }
    else{
        //If we can't get the beads we need
        cout<<"No"<<" ";
        cout<<loss_num<<endl;
    }
    finish=clock();
    //cout<<endl<<"the time cost is:" << double(finish - start) /
CLOCKS_PER_SEC<<endl;
    return 0;
}
```

project3_No_pruning(Test version)

```
#include<iostream>
#include<vector>
#include <fstream>
#include <sys/time.h>
using namespace std;
int goal[256];//Target bead (string)
```

```
string select_input[10001];//Beads from the shop
string goal_string;//Target bead (string)
int select_length;//The length of the target bead
int ans_size=0xfffff;//The final number of extra beads needed
int ans_temp=0;
int judge=1;
int finish=0;
int loss_judge=1;
int loss_num=0;//The number of missing beads
vector<int>ans_num;
void dfs(int index){
    if(index==select_length+1){
        //reache the last string of beads. return
        if(finish==0&&loss_judge==1){
            loss_judge=0;
            for(int i=0;i<256;i++){
                if(goal[i]>0){
                    loss_num+=goal[i];
                }
            }
        }
        return;
    }
    judge=1;
    for(int i=0;i<256;i++){//Determine whether the target bead string has been assembled
        if(goal[i]>0){
            judge=0;
        }
    }
    if(judge==1&&ans_temp<ans_size){
        //If it's already done, and the number of beads needded is less than last time,
        //Reset ans_size
        ans_size=ans_temp;
        finish=1;
    }
    for(int i=0;i<select_input[index].length();i++){
        //Traverse the current bead string
        //and reduce the corresponding number of the target bead string
        int temp=select_input[index][i];
        if(goal[temp]<=0){
            ans_temp++;
        }
        goal[temp]--;
    }
    ans_num.push_back(index);
    dfs(index+1);//Search for the next bead string
    for(int i=0;i<select_input[index].length();i++){
        //Traverse the current bead string
        //and add the corresponding number of the target bead string
        int temp=select_input[index][i];
```

```
        if(goal[temp]<0){
            ans_temp--;
        }
        goal[temp]++;
    }
    ans_num.pop_back();
    dfs(index+1);//Search for the next bead string
}
void initial(){//Initialize all the data used
    for(int i=0;i<256;i++){
        goal[i]=0;
    }
    ans_size=0xfffff;
    ans_temp=0;
    judge=1;
    finish=0;
    loss_judge=1;
    loss_num=0;
}
int main(){
    struct timeval t1,t2,t3,t4;//Define timestamp
    double timeuse,maxtime=-1,mintime=0xfffff;
    gettimeofday(&t1,NULL);//Get the current time
    int times;
    ifstream afile;
    freopen("rand_test.txt","r",stdin );//open rand_test for data reading
    cin>>times;
    for(int ii=0;ii<times;ii++){
        gettimeofday(&t3,NULL);//Get the current time
        initial();//Initialize all the data used
        cin>>goal_string>>select_length;
        for(int i=0;i<select_length;i++){
            select_input[i]=" ";//Initialize all the data used
        }
        for(int i=0;i<goal_string.length();i++){
            goal[goal_string[i]]++;//input the target bead string
        }
        for(int i=0;i<select_length;i++){
            cin>>select_input[i];//input the bead strings in the shop
        }
        dfs(0);//Conduct the search
        if(ans_size<0xfffff){
            //If we can get the beads we need
            cout<<"Yes"<<" ";
            cout<<ans_size<<endl;
        }
        else{
            //If we can't get the beads we need
            cout<<"No"<<" ";
            cout<<loss_num<<endl;
        }
        gettimeofday(&t4,NULL);//Get the current time
```

```
        timeuse = (t4.tv_sec - t3.tv_sec) + (double)(t4.tv_usec - t3.tv_usec)/1000000.0;
        if(timeuse>maxtime){
            maxtime=timeuse;
        }
        if(timeuse<mintime){
            mintime=timeuse;
        }
    }
    gettimeofday(&t2,NULL);//Get the current time
    ////Get the current time
    timeuse = (t2.tv_sec - t1.tv_sec) + (double)(t2.tv_usec - t1.tv_usec)/1000000.0;
    cout<<endl<<"the average time cost is:" << timeuse/double(times)<<endl;
    cout<<endl<<"the max time cost is:" << maxtime<<endl;
    cout<<endl<<"the min time cost is:" << mintime<<endl;
    system("pause");
    return 0;
}
```

project3_with_pruning(Test version)

```
#include<iostream>
#include<vector>
#include<algorithm>
#include <sys/time.h>
#include <fstream>
using namespace std;
int goal[256];//Target bead (string)
int current_bread[256];//The number of characters in the current string
int remain_bread[101][256];//Number of remaining characters in the string
int sell_bread[101][256];//Beads for sale
int value[101];//Record the number of beads that are the same as the destination bead string
string select_input[10001];//Beads from the shop
string goal_string;//Target bead (string)
int select_length;//The length of the target bead
int ans_size=0xfffff;//The final number of extra beads needed
int ans_temp=0;
int judge=1;
int finish=0;
int loss_judge=1;
int loss_num=0;//The number of missing beads
vector<int>ans_num;
void dfs(int index){
    if(ans_temp>ans_size){
        //If the number of beads currently purchased is already greater
        //than the number of additional beads purchased last time, return directly
        return ;
    }
    if(ans_size==0){//If there is no extra bead string,retrun
```

```
        return;
    }
    if(index==select_length+1){
        if(finish==0&&loss_judge==1){
            //When the last string of beads is first found, count the number of missing
beads
                loss_judge=0;
                finish=1;
                //Only calculate the first time, and then do not calculate again
                for(int i=0;i<256;i++){
                    if(goal[i]>0){
                    loss_num+=goal[i];
                    }
                }
            }
        return;//reache the last string of beads. return
    }
    for(int i=0;i<256;i++){
        //If you can't get the beads you need, just retrun
        if(goal[i]>remain_bread[index][i]+current_bread[i]&&finish==1){
            return;
        }
    }
    judge=1;
    for(int i=0;i<256;i++){//Determine whether the target bead string has been assembled
        if(goal[i]>0){
            judge=0;
        }
    }
    if(judge==1&&ans_temp<ans_size){
        //If it's already done, and the number of beads needded is less than last time,
        //Reset ans_size
        ans_size=ans_temp;
        finish=1;
    }
    for(int i=0;i<select_input[index].length();i++){
        //Traverse the current bead string
        //and reduce the corresponding number of the target bead string
        int temp=select_input[index][i];
        if(goal[temp]<=0){
            ans_temp++;
        }
        current_bread[temp]++;
        goal[temp]--;
    }
    ans_num.push_back(index);
    dfs(index+1);//Search for the next bead string
    for(int i=0;i<select_input[index].length();i++){
        //Traverse the current bead string
        //and add the corresponding number of the target bead string
        int temp=select_input[index][i];
        if(goal[temp]<0){
```

```
            ans_temp--;
        }
        current_bread[temp]--;
        goal[temp]++;
    }
    ans_num.pop_back();
    dfs(index+1);//Search for the next bead string
}
void initial(){//Initialize all the data used
    for(int i=0;i<256;i++){
        goal[i]=0;
        current_bread[i]=0;
    }
    for(int i=0;i<101;i++){
        for(int j=0;j<256;j++){
            remain_bread[i][j]=0;
            sell_bread[i][j]=0;
        }
    }
        ans_size=0xfffff;
        ans_temp=0;
        judge=1;
        finish=0;
        loss_judge=1;
        loss_num=0;
}
void mysort(){//Sort the bead string by value
    for(int i=0;i<select_length-1;i++){
        for(int j=0;j<select_length-i-1;j++){
            if(value[j]<=value[j+1]){
                int temp=value[j];
                value[j]=value[j+1];
                value[j+1]=temp;
                string temp2=select_input[j];
                select_input[j]=select_input[j+1];
                select_input[j+1]=temp2;
            }
        }
    }
}
int main(){
    struct timeval t1,t2,t3,t4;//Define timestamp
    double timeuse,maxtime=-1,mintime=0xfffff;
    gettimeofday(&t1,NULL);//Get the current time
    int times;
    ifstream afile;
    freopen("rand_test.txt","r",stdin );//open rand_test for data reading
    cin>>times;
    for(int ii=0;ii<times;ii++){
        gettimeofday(&t3,NULL);//Get the current time
        initial();//Initialize all the data used
        cin>>goal_string>>select_length;
```

```
        for(int i=0;i<select_length;i++){
            select_input[i]=" ";//Initialize all the data used
        }
        for(int i=0;i<goal_string.length();i++){
            goal[goal_string[i]]++;//input the target bead string
        }
        for(int i=0;i<select_length;i++){
            cin>>select_input[i];//input the bead strings in the shop
        }
        for(int i=0;i<select_length;i++){
            for(int j=0;j<select_input[i].size();j++){
                if(goal[select_input[i][j]]){
                //if the character of the bead string is needed
                    value[i]++;
                }
            }
        }
        //mysort();//Sort the bead strings by value
        for(int i=0;i<select_length;i++){
            for(int j=0;j<select_input[i].size();j++){
                //input each character of the optional bead string
                sell_bread[i][select_input[i][j]]++;
            }
        }
        for (int i=select_length-1;i>=0;i--) {
        for (int j=0;j<256;j++)
         //input the number of characters for the remaining bead string
        remain_bread[i][j]=remain_bread[i+1][j]+sell_bread[i][j];
      }
        dfs(0);//Conduct the search
        if(ans_size<0xfffff){
            //If we can get the beads we need
            cout<<"Yes"<<" ";
            cout<<ans_size<<endl;
        }
        else{
            //If we can't get the beads we need
            cout<<"No"<<" ";
            cout<<loss_num<<endl;
        }
        gettimeofday(&t4,NULL);//Get the current time
        timeuse = (t4.tv_sec - t3.tv_sec) + (double)(t4.tv_usec - t3.tv_usec)/1000000.0;
        if(timeuse>maxtime){
            maxtime=timeuse;
        }
        if(timeuse<mintime){
            mintime=timeuse;
        }
    }
    gettimeofday(&t2,NULL);//Get the current time
    ////Get the current time
    timeuse = (t2.tv_sec - t1.tv_sec) + (double)(t2.tv_usec - t1.tv_usec)/1000000.0;
```

```
    cout<<endl<<"the average time cost is:" << timeuse/double(times)<<endl;
    cout<<endl<<"the max time cost is:" << maxtime<<endl;
    cout<<endl<<"the min time cost is:" << mintime<<endl;
    system("pause");
    return 0;
}
```

Samples_product

```
#include <iostream>
#include <vector>
#include <map>
#include <string>
#include<algorithm>
#include <thread>
#include <chrono>
#include<map>
#include <cstdlib>
#include <fstream>
using namespace std;
int main(){
    ofstream oFile;
    oFile.open("rand_test.txt",ios::out);//Open the rand_test.txt for writing
    cout<<"please input the times of tests: ";
    int times;
    cin>>times;//input the number of tests
    oFile<<times;
    oFile<<endl;
    cout<<"please input the number of beads in the shop: ";
    int num;
    cin>>num;//input the number of beads in the shop in every test
    srand((unsigned)time(NULL));
    for(int ii=0;ii<times;ii++){
        int input_length=rand()%1000+1;//Generate a random bead length(1~1000)
        for(int i=0;i<input_length;i++){
            int type=rand()%3;//Generate a random number to determine the type of
character
            int rand_char;
            if(type==0){//Generated number
                rand_char=(rand() % (90-65+1))+ 65;
            }
            else if(type==1){//Generate lowercase letters
                rand_char=(rand() % (57-48+1))+ 48;
            }
            else if(type==2){//Generate capital letters
                rand_char=(rand() % (122-97+1))+ 97;
            }
            oFile<<char(rand_char);
            cout<<char(rand_char);
        }
```

```
        oFile<<endl;
        cout<<endl;
        oFile<<num<<endl;
        cout<<num<<endl;
        for(int i=0;i<num;i++){
            int string_length=rand()%1000+1;//Generate a random bead length(1~1000)
            for(int j=0;j<string_length;j++){
                int type=rand()%3;//Generate a random number to determine the type of character
                int rand_char;
                if(type==0){//Generated number
                    rand_char=(rand() % (90-65+1))+ 65;
                }
                else if(type==1){//Generate lowercase letters
                    rand_char=(rand() % (57-48+1))+ 48;
                }
                else if(type==2){//Generate capital letters
                    rand_char=(rand() % (122-97+1))+ 97;
                }
                oFile<<char(rand_char);
                cout<<char(rand_char);
            }
            oFile<<endl;
            cout<<endl;
        }
    }
    cout<<endl;
    oFile<<endl;
    oFile.close();
}
```