Zadanie Abc – LOGIA 17 (2016/17), etap 3

Treść zadania

Jaś przygotował naszyjnik dla Małgosi. Małgosia poprosiła go o usunięcie niektórych koralików tak, aby wszystkie niebieskie koraliki były przed czerwonymi i zielonymi, a wszystkie czerwone – przed zielonymi. Pozostałe koraliki mogą występować w dowolnych miejscach.

Napisz jednoparametrową funkcję **abc**, której wynikiem jest minimalna liczba koralików do usunięcia. Parametrem funkcji jest niepuste słowo złożone z małych liter alfabetu łacińskiego opisujące kolory kolejnych koralików w oryginalnym naszyjniku. Kolejne znaki oznaczają kolory: n – niebieski, c – czerwony, z - zielony, a pozostałym literom przypisane są inne kolory. Liczba koralików jest nie większa niż 10 000.

Przykłady:

wynikiem **abc('nncnnbffbbbccczzzcz')** jest **2** (wystarczy usunąć trzeci i przedostatni koralik),

wynikiem **abc('zzzznnnnz')** jest **4** (wystarczy usunąć wszystkie niebieskie koraliki).

Omówienie rozwiązania

Zadanie można sprowadzić do policzenia długości najdłuższego ciągu koralików, który spełnia warunki zadania. Taki ciąg będziemy znajdować kolejno przeglądając koraliki, przy czym trzeba pamiętać najdłuższy możliwy ciąg koralików złożony tylko z niebieskich koralików `max_n`, z niebieskich i czerwonych `max_c` i z niebieskich, czerwonych oraz zielonych – `max_z`. W pętli w każdym z kroków:

- Gdy napotkamy koralik inny niż niebieski, czerwony lub zielony, zwiększamy wszystkie trzy liczniki.
- Gdy napotkamy koralik niebieski, zwiększamy licznik niebieskich, pozostałych nie zwiększamy, bo po czerwonych i zielonych, nie mogą występować niebieskie.
- Gdy napotkamy koralik czerwony, obliczamy nową wartość licznika czerwono-niebieskich (`max_c`) jako maksimum niebieskich i czerwonych zwiększony o 1. Może być sytuacja taka, że ciąg niebieskich koralików, będzie teraz ciągiem niebieskich i czerwonych.
- Gdy napotkamy zielony koralik, znajdujemy maksimum z ciągu koralików niebieskich, niebieskich i czerwonych oraz niebieskich – czerwonych – zielonych, zwiększony o 1.

Wynikiem zadania będzie różnica długości wejściowego ciągu i największej z wyliczonych liczb.

Warto zauważyć, że opisana metoda rozwiązania zadania jest dynamicznym podejściem. W programowaniu dynamicznym rozbijamy problem na podproblemy i na podstawie prostszego rozwiązania wnioskujemy o trudniejszym. Przedstawione rozwiązanie jest algorytmem o złożoności liniowej – czas działania algorytmu zależy proporcjonalnie od rozmiaru danych.

Rozwiązanie w języku Python

```python
def abc(napis):
    max_n = 0
    max_c = 0
    max_z = 0
    for e in napis:
        if e not in "ncz":
            max_n += 1
            max_c += 1
            max_z += 1
        else:
            if e == "n":
                max_n += 1
            if e == "c":
                max_c = 1 + max(max_n, max_c)
            if e == "z":
                max_z = 1 + max(max_n, max_c, max_z)

    return len(napis) - max(max_n, max_c, max_z)
```

Testy

Testy obejmują przypadki brzegowe, o danych wygenerowanych losowo dla zwiększającej się długości napisu – 20, 100, 1000 oraz regularne dla dużych danych. Warto zauważyć, że opisany wyżej algorytm daje wyniki w stosunkowo krótkim czasie dla wszystkich testów. Jeśli jednak rozwiązanie będzie w czasie kwadratowym lub wielomianowym, dla dużych testów czas istotnie się wydłuży.

Wywołanie - Python	**Wynik**
abc("n")	0
abc("zcc")	1
abc("zzz")	0
abc("nznzccczzzz")	2
abc("qwerty")	0
abc("ncnznqnnncccnyznnzcn")	7
abc("ncczqczunnczncccznyzbzzcgognnznanozmjzbzzncnnzwzqzcznyciztnzcvqzznce zccnztknzcnznncnnczzznnczpkngzzcln")	40
abc("yczznvlzkzzwzfcncsnznnznznxznzcpznckннknnzclzccjccccnzccccnvnonznnnzh cnzcdnrnccqcnhcindczccnnonozzczczzcnzjwnzznccznnzzzncuncqznusfczcnbzznnnz cbzznceceзindzccncnczclczzcozzjnczmcczcbcnrzzcnnfcnnezjznhzdennnczctpxvnzzz nnzanzcccnzznnnzcczcnxznczjenzznnnvzhcznznnzznnznnznngzccncnnnczccnzwnz cbcnncncncnfnccpcczcnvhcznanzcuwcccnzzzwncbcznbczedxccnzznzbzrzzczcnccnu cnnncacczzzznncapnnnzbcnrcncicennncnnzznczncnfznzncnevfbcznznzknnzccncznhcc zzdzlcnnzenccvcngcimynzpzpnzndycpczczcnvcczzczcxrjnvzznzzcnnnnczznlkncncnc zcocqznccncznxzhcnndftnyzcadccbchfacctcncnccnznnzzzqnczzznczzfcczzzcnsce bcbcrnznznmzzcznccznzczccdczyntcnzszocxnnnzcnfyczeeekccnzzanzcnncjzznac cczcccpzzczmzczddrcwnnnczcznvnnncnynzcgcinncnyrnzcccznnlnnzcccpzzccnzzjz nelcnqzcnnzwnzzdznntcnczzzccczccchzzzczrznfznzawcmzocnpcczznhcniztcznqzz czrynznzpjtcczvnnwzzczccncncyncgccmhnznzcncnzeznznzzfccqcnnncxnzzcnnncc cnnznczznzitnzcccdcnzecnnzzqncdcccknnkzzcczenclzxhcnczzznzzzcehnnnnnonntz nccnczorzcozzcznnnnzdlnzcnnzznocicctsncnnkczczonzzznczz")	486
abc("n"*3333+"c"*3333+"z"*3333+"c")	1
abc("ncz"*3333)	6664

Zadanie Litera – LOGIA 17 (2016/17), etap 3

Treść zadania

Napisz jednoparametrową funkcję **litera**, której parametrem jest niepusta lista słów składających się z małych liter alfabetu łacińskiego. Wynikiem funkcji jest ta litera, która występuje najczęściej. Jeśli jest więcej niż jedna taka litera, to wynikiem funkcji jest uporządkowana rosnąco lista liter mających tę własność.

Przykłady:

wynikiem **litera(['ala', 'ma', 'kota'])** jest **'a'**

wynikiem **litera(['julka', 'lubi', 'psy'])** jest **['l', 'u']**

Omówienie rozwiązania

Jest to zadanie typowo narzędziowe, związane z przetwarzaniem napisów. Nie wymaga wiedzy algorytmicznej, poza zliczaniem wystąpień elementów i poszukiwaniem maksimum. Ponieważ mamy 26 liter alfabetu łacińskiego należy na początku utworzyć listę 26 wyzerowanych liczników. Następnie przeglądamy w pętli poszczególne wyrazy, a dla konkretnego wyrazu jego litery i powiększamy odpowiednie liczniki. Następnie należy znaleźć maksimum na liście liczników i policzyć, ile razy ono występuje. Jeśli raz, to wynikiem zadania jest litera odpowiadająca tej wartości (indeks listy zamieniony na literę jej odpowiadającą). Natomiast, jeśli ta sama wartość maksimum występuje więcej razy na liście, należy przejrzeć liczniki i utworzyć listę z literami odpowiadającymi wartościom maksimum.

Alternatywnym rozwiązaniem jest połączenie wyrazów w jeden napis i zliczanie liter występujących w napisie. Rozważania dotyczące znajdowania maksimum pozostają bez zmian.

Rozwiązanie w języku Python

```
1.  def litera(slowa):
2.      liczniki = [0 for i in range(26)]
3.      for slowo in slowa:
4.          for litera in slowo:
5.              liczniki[ord(litera)-97] += 1
6.      m = max(liczniki)
7.      if liczniki.count(m) == 1:
8.          return chr(97+liczniki.index(m))
9.      else:
10.         wynik = []
11.         for i in range(26):
12.             if liczniki[i] == m:
13.                 wynik.append(chr(97+i))
14.         return wynik
```

W wierszu 2 generowana jest lista 26 liczników z wyzerowaną wartością początkową. Pętla w wierszu 3 przegląda słowa listy słów, a wewnętrzna pętla w wierszu 4 litery pojedynczego słowa. W wierszu 5 powiększany jest o jeden licznik odpowiadający danej literze. Ponieważ lista jest indeksowana od 0, od kodu ASCII litery odejmowane jest 97 (kod ASCII małej litery a). W wierszu 6 wyznaczana jest wartość maksimum (wykorzystujemy funkcję max w Pythonie). Jeśli wartość maksimum występuje tylko raz na liście (funkcja count), to wynikiem jest litera o kodzie ASCII równym indeksowi maksimum

powiększonemu o 97. W przeciwnym przypadku budujemy listę liter. Na początku jest ona pusta, następnie w pętli dodawane są kolejne litery, których liczba wystąpień jest równa maksimum. Są one od razu uporządkowane alfabetycznie.

Testy

Należy przeprowadzić testy zarówno dla sytuacji, gdy wynikiem jest pojedyncza litera, jak lista liter. Warto wykonać test, gdy poprawnym wynikiem jest cały alfabet. Słowa do testów powinny być zarówno krótkie (w szczególności jednoliterowe) jak i długie. Także liczba słów w testach powinna być różna. Przykładowe testy:

```
litera(['abc', 'bac', 'cab'])
```

wynik: `['a', 'b', 'c']`

```
litera(['bacdefghijklmnopqrstuvwxyz', 'aeouy', 'zyxwvutsrqponmlkjihgfedcab', 'diefghij'])
```

wynik: `['e', 'i']`

```
litera(['cabdefghijklmnopqrstuvwxyz','zyxwvutsrqponmlkjihgfedbac','acegikmoqsuwybdfhjlnprtvxz'])
```

wynik: `['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z']`

```
litera(['d', 'b', 'c', 'a', 'ddd', 'eeu', 'eu', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'fug', 'hiju', 'klmno', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z'])
```

wynik: `'u'`

Zadanie Planeta – LOGIA 17 (2016/17), etap 3

Treść zadania

Na pewnej planecie domy mają adresy będące parami liczb całkowitych dodatnich – ich współrzędnych. Rozmiar planety to maksymalna możliwa wartość współrzędnej. Na planecie o rozmiarze N jest N x N adresów, od (1, 1) do (N, N).

Po planecie można poruszać się tylko w kierunkach północ↔południe oraz wschód↔zachód (nie po skosie), także w kółko. Odległość między domami jest sumą minimalnych różnic współrzędnych. Na przykład dla planety o rozmiarze 10 dom o adresie (6, 7) jest oddalony od domu o adresie (7, 9) o 1 + 2 = 3, a dom o adresie (1, 4) jest oddalony od domu o adresie (8, 9) o 3 + 5 = 8, a nie 7 + 5 = 12.

Na planecie wyróżniamy osiedla, tj. rozłączne zbiory domów. Osiedla mogą składać się z jednego lub większej liczby domów. Dom należy do osiedla, gdy jego odległość od pewnego domu tego osiedla jest nie większa niż 5. Jeśli odległość od danego domu do każdego innego na planecie jest większa niż 5, to taki dom stanowi jednodomowe osiedle.

Napisz dwuparametrową funkcję **planeta**, której pierwszym parametrem jest rozmiar planety, a drugim lista adresów domów. Wynikiem jest liczba domów w osiedlu składającym się z największej liczby domów. Na planecie stoją co najmniej 2 domy i nie więcej niż 5 000. Rozmiar planety jest nie mniejszy niż 2 i nie większy niż 5 000. Adresy domów nie powtarzają się.

Przykłady:

wynikiem **planeta(12, [[3,1], [1,1], [1,3], [2,12], [9,5], [8,6]])** jest **4**,

wynikiem **planeta(100, [[6,6], [6,11], [11,6], [11,11], [80,80]])** jest **4**.

Omówienie rozwiązania

Zadanie będziemy rozwiązywać rozpatrując poszczególne domy. Bierzemy i usuwamy pierwszy dom z listy i wstawiamy go do kolejki - listy pomocniczej `pom`. Następnie do kolejki wstawiamy wszystkich jego sąsiadów – domy, które są oddalone co najwyżej o 5. Postępowanie kontynuujemy dla wszystkich domów z listy `pom`. W ten sposób znajdujemy rozmiar osiedla `pom_b`. Największy dotychczasowy rozmiar zapamiętujemy na zmiennej pomocniczej `max_o`. Po wyczerpaniu się kolejki, bierzemy kolejny dom z listy danych. Wynikiem programu będzie rozmiar największego osiedla, czyli `max_o`.

Zadanie jest przykładem problemu grafowego, w którym znajdujemy spójne składowe. Wierzchołkami grafu są domy, a krawędzie są wyznaczone przez odległość nie większa niż 5.

Pozostaje jeszcze zdefiniować funkcję pomocniczą wyliczającą odległość między domami. Odległość jest sumą różnicy współrzędnych. Przy czym, jeśli różnica jest zwiększa niż połowa rozmiaru planety bierzemy jej dopełnienie.

OEiiZK
Ośrodek Edukacji Informatycznej
i Zastosowań Komputerów w Warszawie

Rozwiązanie w języku Python

Implementacja jest złożona z dwóch funkcji – `planeta` i funkcji pomocniczej `odl`. Przy wyliczaniu wartości bezwzględnej korzystamy z funkcji `abs`. Będziemy też potrzebować różnych funkcji operujących na listach: `append` – dołączanie elementów do listy, `pop` – pobieranie i usuwanie pierwszego, `remove` – usuwanie określonego elementu listy.

```
1.  def odl(a, b, r):
2.      x = abs(a[0]-b[0])
3.      if x > r/2:
4.          x = r - x
5.      y = abs(a[1]-b[1])
6.      if y > r/2:
7.          y = r - y
8.      return x + y
9.
10. def planeta(r, lista):
11.     max_o = 0
12.     while len(lista) > 0:
13.         pom = []
14.         max_b = 1
15.         pom.append(lista.pop())
16.         while len(pom) > 0:
17.             e = pom.pop()
18.             # rozpatrujemy wszystkich sąsiadów
19.             for s in lista:
20.                 if odl(e, s, r) <= 5:
21.                     lista.remove(s)
22.                     pom.append(s)
23.                     max_b += 1
24.         if max_b > max_o:
25.             max_o = max_b
26.     return max_o
```

Testy

Testy obejmują proste przypadki, która można przeanalizować rysując graf na papierze oraz większe, by sprawdzić różne sytuacje – domy są położone gęsto lub rzadko, osiedla znajdują się na środku lub mamy do czynienia z zawijaniem.

```
planeta(2,[[1,1],[2,2]])
```
wynik: 2

```
planeta(10,[[1,1],[6,6],[6,7],[7,6]])
```
wynik: 3

```
planeta(30,[[6,6],[6,7],[6,8],[6,9],[6,10],[7,10],[8,10],[9,10],[10,10],[11,10],[12,10]
,[13,10],[14,10],[15,10],[16,6],[16,7],[16,8],[16,9],[16,10],[16,11],[16,12],[16,13],[1
6,14],[16,15],[25,25],[26,26]])
```
wynik: 24

```
planeta(50,[[6,6],[6,7],[6,8],[6,9],[6,10],[7,10],[8,10],[9,10],[10,10],[11,10],[12
,10],[13,10],[14,10],[15,10],[16,7],[16,8],[16,9],[16,10],[16,11],[16,12],[16,13],[
16,14],[16,15],[7,6],[8,6],[9,6],[10,6],[11,6],[12,6],[13,6],[14,6],[15,6],[16,6],[
17,10],[18,10],[19,10],[20,10],[21,10],[22,10],[23,10],[24,10],[25,10],[26,7],[26,8
],[26,9],[26,10],[26,11],[26,12],[26,13],[26,14],[26,15],[17,6],[18,6],[19,6],[20,6
],[21,6],[22,6],[23,6],[24,6],[25,6],[26,6],[30,28],[30,29],[30,30]])
```
wynik: 61

```
planeta(20,[[7,1],[7,20],[7,19],[7,2],[8,2],[9,2],[10,2],[11,2],[12,2],[13,2],[14,2
],[15,2],[16,2],[17,2],[17,1],[17,20],[17,19],[10,10]])
```
wynik: 17

```
planeta(100,[[91,46], [68,34], [16,28], [21,29], [72,74], [59,85], [36,83], [94,45], [74,75],
[48,4], [64,88], [76,63], [38,2], [55,86], [44,3], [26,79], [27,27], [22,41], [33,36], [64,45],
[79,4], [29,37], [49,28], [94,90], [90,75], [26,29], [81,14], [96,3], [10,71], [32,7], [93,70],
[74,45], [65,18], [26,47], [85,19], [34,47], [61,93], [54,99], [53,78], [31,77], [100,49],
[73,18], [38,35], [76,82], [26,5], [53,36], [82,60], [52,89], [91,65], [75,41], [52,85],
[40,18], [71,22], [34,96], [8,71], [54,49], [45,33], [46,79], [34,57], [71,98], [28,91],
[65,68], [24,87], [80,43], [57,62], [7,42], [24,22], [31,66], [39,12], [72,26], [32,29],
[90,42], [53,19], [84,57], [47,55], [1,40], [2,54], [28,90], [87,80], [55,73], [35,55],
[100,33], [49,53], [65,73], [11,75], [54,4], [39,6], [49,89], [26,15], [68,36], [45,53],
[13,30], [1,39], [4,97], [78,69], [48,31], [12,64], [85,52], [62,49], [28,52], [93,39], [67,54],
[80,51], [17,6], [85,100], [28,60], [83,16], [82,25], [39,10], [55,88], [46,73], [9,48],
[88,77], [44,98], [89,25], [31,50], [60,54], [100,6], [49,11], [56,92], [60,22], [47,32],
[6,59], [7,12], [96,54], [46,75], [76,13], [64,40], [77,54], [11,25], [45,40], [13,48], [47,92],
[43,98], [74,24], [5,30], [98,2], [61,9], [49,38], [16,14], [17,75], [42,76], [93,99], [14,5],
[7,25], [70,48], [33,100], [39,58], [56,44], [88,78], [26,26], [79,83], [54,10], [34,58],
[60,84], [8,100], [12,14], [61,41], [39,23], [22,89], [59,37], [47,56], [1,25], [50,19],
[28,99], [14,49], [71,99], [19,63], [64,10], [74,65], [21,8], [41,75], [24,24], [83,43],
[92,60], [77,15], [66,21], [66,4], [16,27], [59,51], [58,23], [48,8], [66,34], [80,40], [69,55],
[5,8], [52,66], [49,46], [3,2], [7,61], [43,46], [30,86], [64,85], [30,77], [76,90], [74,21],
[26,57], [56,23], [1,42], [44,93], [11,66], [45,34], [42,62], [59,29], [100,13], [92,55],
[6,91], [44,78], [98,87], [13,17], [57,9], [75,70], [63,79], [17,40], [43,60], [46,15], [79,82],
[99,73], [73,68], [29,30], [96,66], [32,55], [63,32], [49,98], [19,41], [64,24], [24,100],
[60,55], [61,99], [29,38], [21,28], [31,13], [41,34], [55,42], [75,13], [33,13], [49,45],
[87,57], [27,20], [40,7], [64,25], [6,9], [62,77], [30,70], [23,24], [66,78], [81,57], [93,52],
[54,96], [25,92], [38,44], [48,51], [59,31], [97,37], [25,77], [64,21], [61,61], [92,12],
[19,71], [89,15], [94,55], [80,5], [82,29], [81,53], [65,89], [49,8], [20,30], [54,23], [64,98],
[75,57], [99,94], [19,18], [24,18], [98,10], [9,36], [40,1], [54,92], [99,45], [95,98], [92,95],
[11,78], [91,5], [41,59], [29,17], [38,19], [14,61], [2,35], [52,31], [60,8], [28,10], [8,47],
[89,43], [25,28], [27,8], [9,84], [77,17]])
```
wynik: 3

Zadanie Zawody – LOGIA 17 (2016/17), etap 3

Treść zadania

Zawodnicy (co najmniej dwóch, co najwyżej dwudziestu sześciu) przed startem otrzymują identyfikatory oznaczone kolejnymi wielkimi literami alfabetu łacińskiego (A, B, C, ... itd.), których nie zmieniają podczas zawodów. Zawody składają się z kolejnych rund. Sekwencja zawodników w kolejnych rundach zostaje zachowana, ale zawodników ubywa. Po każdej rundzie odpada zawodnik, który w danej rundzie uzyskał najmniej punktów. Wyniki każdej rundy zapisywane są w liście zawierającej liczby punktów uzyskanych przez kolejnych zawodników w tej rundzie (zakładamy, że każdy zawodnik ma inną liczbę punktów).

Napisz jednoparametrową funkcję **zawody**, której parametrem jest poprawna lista opisująca przebieg zawodów, tj. lista wyników kolejnych rund. Wynikiem funkcji jest litera oznaczająca zawodnika, który zwyciężył, czyli wygrał ostatnią rundę.

Przykłady:

wynikiem **zawody([[8, 9]])** jest **'B'**,

wynikiem **zawody([[4, 0, 2, 1], [1, 2, 3], [2, 1]])** jest **'C'** (po pierwszej rundzie odpadł zawodnik B, po drugiej A, a w ostatniej lepszy był C niż D).

Omówienie rozwiązania

Długość listy opisującej wyniki pierwszej rundy (pierwszy element listy opisującej zawody) wyznacza liczbę zawodników. Warto utworzyć pomocniczą listę składającą się z identyfikatorów startujących zawodników, czyli *n* początkowych wielkich liter alfabetu (gdzie n oznacza długość listy z wynikami pierwszej rundy). Rozwiązanie zadania polega na przejrzeniu listy z przebiegiem zawodów i dla każdej rundy znalezienie pozycji minimum. Z pomocniczej listy z identyfikatorami zawodników należy usunąć zawodnika na znalezionej pozycji, czyli z najgorszym wynikiem w danej rundzie. Po przejrzeniu wyników wszystkich rund pomocnicza lista będzie zawierała jeden element, identyfikator zawodnika, który wygrał zawody. Pierwszy element tej listy jest wynikiem.

OEiiZK
Ośrodek Edukacji Informatycznej
i Zastosowań Komputerów w Warszawie

Rozwiązanie w języku Python

```
1.  def zawody(wyniki):
2.      n = len(wyniki[0])
3.      zawodnicy = [chr(i+65) for i in range(n)]
4.      for wynik in wyniki:
5.          m = min(wynik)
6.          i = wynik.index(m)
7.          del zawodnicy[i:i+1]
8.      return zawodnicy[0]
```

W wierszu 2 w zmiennej n zostaje zapamiętana liczba zawodników. W następnym wierszu generowana jest lista n początkowych wielkich liter alfabetu łacińskiego (kody ASCII wielkich liter rozpoczynają się od 65). W pętli w wierszach 4-7 przeglądana jest lista z wynikami kolejnych rund. Zmienna *m* określa najgorszy wynik danej rundy (minimalny element listy z wynikami rundy, korzystamy z funkcji `min`), a zmienna *i* indeks minimum. W wierszu 7 zawodnik jest usuwany z listy zawodników uczestniczących jeszcze w zawodach. Notacja `z :` dla polecenia `del` określa pierwszy usuwany element i pierwszy, który pozostanie na liście.

Testy

Należy przeprowadzić testy dla różnej liczby zawodników od 2 do 26. Wyniki punktowe nie mają znaczenia. Przykładowe testy:

Wywołanie – Python	**Wynik**
zawody([[4, 1, 3, 2], [5, 6, 7], [9, 8]])	'C'
zawody([[5, 4, 3, 2, 1], [4, 3, 2, 1], [3, 2, 1], [1, 2]])	'B'
zawody([[1, 2, 3, 4, 5, 6], [1, 2, 3, 4, 5], [1, 2, 3, 4], [2, 1, 3], [1, 2]])	'F'
zawody([[11, 1, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 9], [11, 13, 2, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 9], [11, 13, 15, 3, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 9], [11, 13, 15, 17, 4, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 9], [11, 13, 15, 17, 19, 5, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 9], [11, 13, 15, 17, 19, 21, 6, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 9], [11, 13, 15, 17, 19, 21, 23, 7, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 9], [11, 13, 15, 17, 19, 21, 23, 25, 8, 27, 28, 29, 30, 31, 32, 33, 34, 35, 9], [11, 13, 15, 17, 19, 21, 23, 25, 27, 9, 29, 30, 31, 32, 33, 34, 35, 11], [11, 13, 15, 17, 19, 21, 23, 25, 27, 29, 30, 31, 32, 33, 34, 35, 1], [11, 13, 15, 17, 19, 21, 23, 25, 27, 29, 30, 31, 32, 33, 2, 35], [11, 13, 15, 17, 19, 21, 23, 25, 27, 29, 30, 31, 3, 33, 35], [11, 13, 15, 17, 19, 21, 23, 25, 27, 29, 4, 31, 33, 35], [1, 13, 15, 17, 19, 21, 23, 25, 27, 29, 31, 33, 35], [13, 15, 2, 19, 21, 23, 25, 27, 29, 31, 33, 35], [13, 15, 19, 21, 3, 25, 27, 29, 31, 33, 35], [13, 15, 19, 21, 25, 27, 4, 31, 33, 35], [13, 15, 19, 21, 25, 27, 31, 33, 5], [6, 4, 9, 8, 5, 7, 1, 3], [4, 6, 9, 2, 5, 7, 3], [3, 5, 9, 7, 6, 4], [1, 9, 3, 7, 5], [9, 5, 7, 2], [1, 7, 5], [5, 7]])	'Q'