Project 6

Cæcilia, Lea & Peter

Folding method

1/4 approximation algorithm

Finding evens and odds

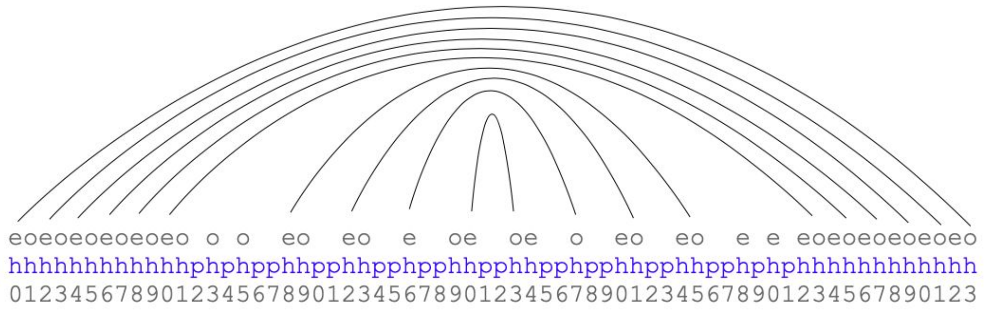
```python
def evens_odds(S):
    evens_odds = []
    pointer = 0
    for i in S:
        if i == 'h' and pointer % 2 == 0:
            evens_odds.append('e')
        if i == 'h' and pointer % 2 == 1:
            evens_odds.append('o')
        elif i != "h":
            evens_odds.append(" ")
        pointer += 1
    return evens_odds
```

```
eoeoeoeoeoeo o o  eo  eo  e  oe  oe  o  eo  eo  e e eoeoeoeoeoeo
hhhhhhhhhhhhphphpphhpphhpphpphhpphhpphpphhpphhpphphphhhhhhhhhhhh
0123456789012345678901234567890123456789012345678901234567890123
```

[’e’, ’o’, ’e’, ’o’, ’e’, ’o’, ’e’, ’o’, ’e’, ’o’, ’e’, ’o’, ’ ’, ’o’, ’ ’, ’o’, ’ ’, ’ ’, ’e’, ’o’, ’ ’, ’ ’, ’e’, ’o’, ’ ’, ’ ’, ’e’, ’ ’, ’ ’, ’o’, ’e’, ’ ’, ’ ’, ’o’, ’e’, ’ ’, ’ ’, ’o’, ’ ’, ’ ’, ’e’, ’o’, ’ ’, ’ ’, ’e’, ’o’, ’ ’, ’ ’, ’e’, ’ ’, ’e’, ’ ’, ’e’, ’o’, ’e’, ’o’, ’e’, ’o’, ’e’, ’o’, ’e’, ’o’, ’e’, ’o’]

Match evens from left with odds from right

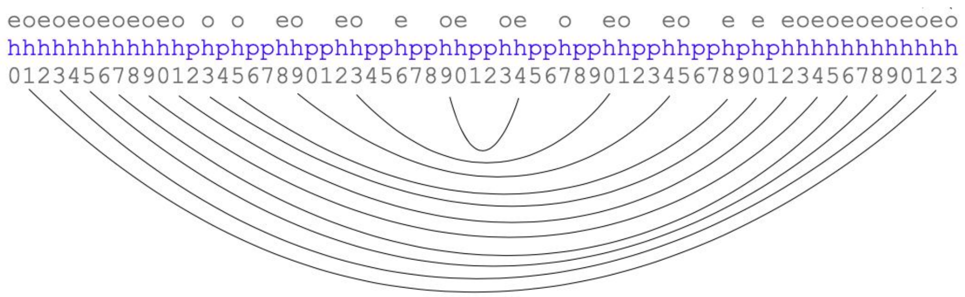
```python
def match_evens_from_left(S):
    e_o_list = evens_odds(S)
    matches = []
    i = 0
    j = len(e_o_list)-1
    while i < 0.5 * len(e_o_list) and j > 0.5 * len(e_o_list):
        if e_o_list[i] == 'e' and e_o_list[j] == 'o':
            matches.append((i, j))
            i += 1
            j -= 1
        if e_o_list[i] == 'e' and e_o_list[j] != 'o':
            j -= 1
        if e_o_list[i] != 'e' and e_o_list[j] == 'o':
            i += 1
        if e_o_list[i] != 'e' and e_o_list[j] != 'o':
            i += 1
            j -= 1
    return matches
```


eoeoeoeoeoeo o o eo eo e oe oe o eo eo e e eoeoeoeoeoeo
hhhhhhhhhhhhphphpphhpphhpphpphhpphhpphpphhpphhpphphphhhhhhhhhhhh
0123456789012345678901234567890123456789012345678901234567890123

[(0, 63), (2, 61), (4, 59), (6, 57), (8, 55), (10, 53), (18, 45), (22, 41), (26, 37), (30, 33)]

Match odds from left with evens from right

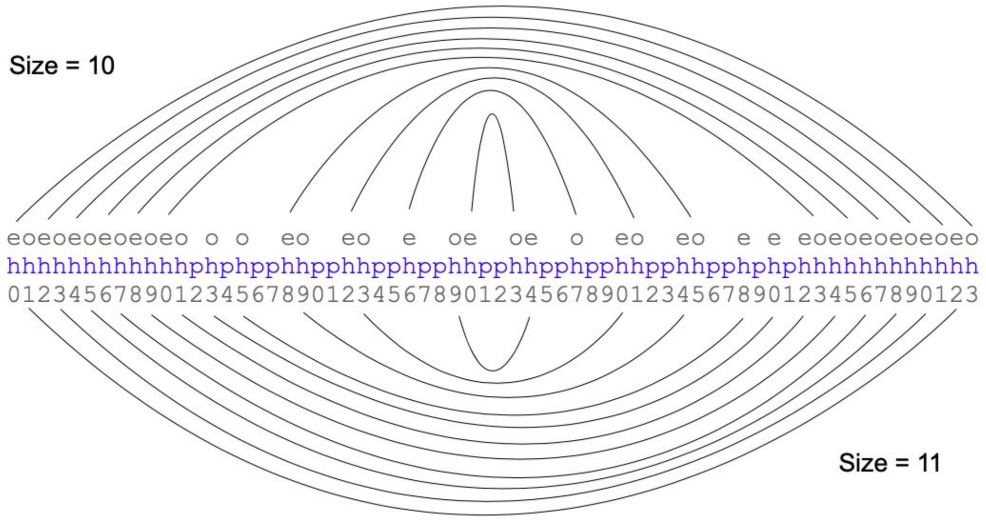
```
def match_odds_from_left(S):
    e_o_list = evens_odds(S)
    matches = []
    i = 0
    j = len(e_o_list)-1
    while i < 0.5 * len(e_o_list) and j > 0.5 * len(e_o_list):
        if e_o_list[i] == 'o' and e_o_list[j] == 'e':
            matches.append((i, j))
            i += 1
            j -= 1
        if e_o_list[i] == 'o' and e_o_list[j] != 'e':
            j -= 1
        if e_o_list[i] != 'o' and e_o_list[j] == 'e':
            i += 1
        if e_o_list[i] != 'o' and e_o_list[j] != 'e':
            i += 1
            j -= 1
    return matches
```


eoeoeoeoeoeo o o eo eo e oe oe o eo eo e e eoeoeoeoeoeo
hhhhhhhhhhhhphphpphhpphhpphpphhpphhpphpphhpphhpphphphhhhhhhhhhhh
0123456789012345678901234567890123456789012345678901234567890123

[(1, 62), (3, 60), (5, 58), (7, 56), (9, 54), (11, 52), (13, 50), (15, 48), (19, 44), (23, 40), (29, 34)]

Pick the maximum sized matching

```python
def max_match(S):
    if len(match_evens_from_left(S)) > len(match_odds_from_left(S)):
        max = match_evens_from_left(S)
    else:
        max = match_odds_from_left(S)
    return max
```

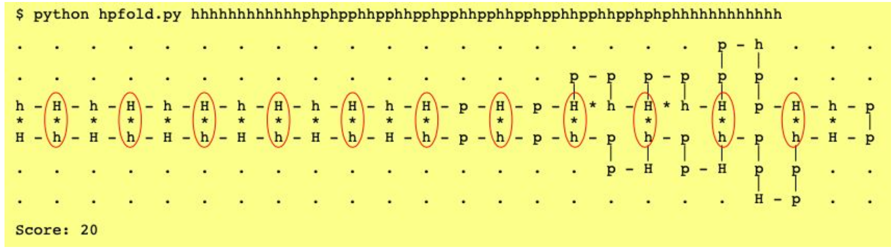


Size = 10
eoeoeoeoeoeo o o eo eo e oe oe o eo eo e e eoeoeoeoeoeo
hhhhhhhhhhhhphphpphhpphhpphpphhpphhpphpphhpphhpphphphhhhhhhhhhhh
0123456789012345678901234567890123456789012345678901234567890123
Size = 11

[(1, 62), (3, 60), (5, 58), (7, 56), (9, 54), (11, 52), (13, 50), (15, 48), (19, 44), (23, 40), (29, 34)]

Make fold from the matching

```python
def hpfold(S):
    forward = max_match(S)
    fold = ""
    i = 0
    fold = fold + (forward[0][0]) * "f"
    while i in range(len(forward)-1):
        D = forward[i+1][0] - forward[i][0]
        K = D - 4
        if D == 2:
            fold = fold + "ff"
        elif D > 2 and D % 2 == 0:
            fold = fold + "l" + (int(K/2) * "f") + "rr" + (int(K/2) * "f") + "l"
        i += 1
    D = forward[-1][1] - forward[-1][0]
    K = D // 2
    fold = fold + (int(K)*"f") + "rr" + (int(K-1)*"f")
    j = len(forward)-1
    while j in range(len(forward)-1, 0, -1):
        D = forward[j-1][1] - forward[j][1]
        K = D - 4
        if D == 2:
            fold = fold + "ff"
        elif D > 2 and D % 2 == 0:
            fold = fold + "l" + (int(K/2) * "f") + "rr" + (int(K/2) * "f") + "l"
        j -= 1
    fold = fold + (len(S)-1 - forward[0][1]) * "f"
    return fold
```



```
$ python hpfold.py hhhhhhhhhhhhphphpphhpphhpphpphhpphhpphpphhpphhpphphphhhhhhhhhhhhh

.   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   p - h   .   .   .
                                                                            |   |
.   .   .   .   .   .   .   .   .   .   .   .   .   .   .   p - p   p - p   p   p   .   .   .
                                                            |   |   |   |   |   |
h - H - h - H - h - H - h - H - h - H - h - H - p - H - p - H * h - H * h - H   p - H - h - p
*   *   *   *   *   *   *   *   *   *   *   *       *       *       *       *       *   *   |
H - h - H - h - H - h - H - h - H - h - H - h - p - h - p - h - p   h - p   h - p   h - H - p
                                                                |   |   |   |   |   |
.   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   p - H   p - H   p   p   .   .
                                                                                |   |
.   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   H - p   .   .

Score: 20
```

fffffffffffffffl rrllrrllfrrflffrrflfrrfllrrllrrlfffffffffffffff

Results

Sequence number	Our score	Optimal score	Ratio	Running time (sec)
1	2	4	0,50	1,149E-04
2	5	8	0,63	9,394E-05
3	4	9	0,44	9,799E-05
4	5	9	0,56	9,966E-05
5	3	10	0,30	1,063E-04
6	4	9	0,44	1,142E-04
7	4	8	0,50	1,318E-04
8	5	14	0,36	8,700E-04
9	15	23	0,65	1,879E-04
10	12	21	0,57	1,347E-03
11	20	36	0,56	1,948E-04
12	20	42	0,48	1,938E-04
13	25	53	0,47	2,789E-04
14	24	48	0,50	3,107E-04
15	22	50	0,44	3,099E-04

Mean			**0,49**	**3,264E-04**