8-12

Let C[i][j] be the cost of breaking a substring from position i to position j into multiple pieces, where the breaks are made after the characters specified in the input list. We want to find the minimum cost of breaking the entire string, which corresponds to the value of C[0][n], where n is the length of the input string.

OPTCOST(0, k+1) = min{n +OPTCOST(0, i) + OPTCOST(i, k+1)}

The base case is when the length of the substring is 1, in which case the cost is 0.

The time complexity of this algorithm is O(n^3), since we have n^2 subproblems, and each subproblem requires O(n) time to compute.

i/j	0	1	2	3	n-1	n
0	0	0	0	0	0	0
1	0	0	0	0	0	0
2		0	0	0	0	0
3			0	0	0	0
n-1				0	0	0
n					0	0

Algorithm pseudo code:

```
for length = 2 to n:

for i = 1 to n - length + 1:

j = i + length - 1

C[i][j] = infinity

for m = i to j - 1:

cost = C[i][m] + C[m+1][j] + (j-i+1)

C[i][j] = min(C[i][j], cost)
```

Traceback step code:

```
def traceback(i, j, k, positions):

if k == 1:

return [i, j]

m = positions[i][j][k]

left = traceback(i, m, k-1, positions)

right = traceback(m+1, j, 1, positions)

return left + right[1:]
```

8-14

g - # games left

i - # points needed for champion to retain title

p(g,i)

if odd plays with black

if even play with white

win p(g-1,i-1)

lose p(g-1,i)

draw p(g-1,i-1/2)

goal: p(24,12)

$$P_{g,i} = w_w\left(P_{g-1,i-1}\right) + w_d\left(P_{g-1,i-\frac{1}{2}}\right) + w_l\left(P_{g-1,i}\right) + b_{w\left(P_{g-1,i-1}\right)} + b_d\left(P_{g-1,i-\frac{1}{2}}\right) + b_l\left(P_{g-1,i}\right)$$

Code:

```
let p[0...g][0...i]

for k = 0 to g:
   p[k][0] = 1.0  // base case

for k = 1 to g:
   for j = 1 to i:
      p[k][j] = ww * bd * p[k-1][j-1] + ww * bl * p[k-1][j-1] + wd * p[k-1][j] +
            bw * wd * p[k-1][j-1] + bw * ww * p[k-1][j-1] + bd * p[k-1][j] +
            bl * ww * p[k-1][j-1] + bl * bd * p[k-1][j-1]
```

Since the algorithm uses a 2D table for size n * n and each entry is constant, the running time of the algorithm is O(n^3)