Gangsta

Calvin Fung

July 19, 2025

1 Problem

You are given a binary string $s_1 s_2 ... s_n$ of length n . A string s is called binary if it consists only of zeros and ones.

For a string p , we define the function $f(p)$ as the maximum number of occurrences of any character in the string p . For example, $f(0010) = 3$, $f(01) = 1$.

You need to find the sum $f(s_l s_{l+1} ... s_r)$ for all pairs $1 \leq l \leq r \leq n$.

2 Solution

We present an $O(n)$ time solution.

2.1 analysis

Let

- $s_{u,v}$ be the substring $s_u s_{u+1} ... s_v$
- a partial sum $P_i = \sum_{j=1}^{i} f(s_{j,i})$. Notice the desired answer $\sum_{1 \leq l \leq r \leq n} f(s_{l,r})$ is $\sum_{i=1}^{n} P_i$, our goal is to compute all P_i.
- a function $d(s_{u,v})$ defined to be equal the number of 1s in $s_{u,v}$ minus the number of 0s in $s_{u,v}$. This value can range from $-n$ to n.
- a value $T_{i,1}$ defined to be equal the number of substrings ending at i that has more 1s than 0s ie. $T_{i,1} = |\{s_{j,i}, j = 1, ..., i : d(s_{j,i}) > 0\}|$
- a value $T_{i,2}$ defined to be equal the number of substrings ending at i that has the same number of 1s and 0s ie. $T_{i,2} = |\{s_{j,i}, j = 1, ..., i : d(s_{j,i}) = 0\}|$
- a value $T_{i,3}$ defined to be equal the number of substrings ending at i that has more 0s than 1s ie. $T_{i,3} = |\{s_{j,i}, j = 1, ..., i : d(s_{j,i}) < 0\}|$

Notice $T_{i,1}, T_{i,2}, T_{i,3}$ partition the set of substrings ending at i $\{s_{j,i}, j = 1, ..., i\}$ into the 3 sets shown above.

Now given $P_i, T_{i,1}, T_{i,2}, T_{i,3}$, we show how to compute $P_{i+1}, T_{i+1,1}, T_{i+1,2}, T_{i+1,3}$ in $O(1)$ time. Below suppose we're dealing with the case that $s_{i+1} = 1$, the other case $s_{i+1} = 0$ is symmetric.

First, we claim $P_{i+1} = P_i + T_{i+1,1} + T_{i+1,2} + 1$. To see why, consider any $s_{j,i+1}$, we have 4 cases :

$j = i + 1$: then $f(s_{j,i+1}) = f(1) = 1$

$d(s_{j,i}) > 0$: then both $f(s_{j,i}), f(s_{j,i+1})$ count the number of 1s and $f(s_{j,i+1}) = f(s_{j,i}) + 1$

$d(s_{j,i}) = 0:$ then both $f(s_{j,i}), f(s_{j,i+1})$ count the number of 1s and $f(s_{j,i+1}) = f(s_{j,i}) + 1$

$d(s_{j,i}) < 0:$ then both $f(s_{j,i}), f(s_{j,i+1})$ count the number of 0s and $f(s_{j,i+1}) = f(s_{j,i})$

now it's easy to see our claim is correct by expanding $P_{i+1} = \sum_{j=1}^{i+1} f(s_{j,i+1})$ and rearranging terms.

Then we discuss how to compute the Ts efficiently. It's not hard to see $T_{i+1,1} = T_{i,1}+T_{i,2}+1$. However, it's not as easy to compute $T_{i+1,2}, T_{i+1,3}$. To do so we need another data structure, let O_i, Z_i be lists such that $O_i[k]$ stores the number of substrings ending at i that has d value equals to k ie. $O_i[k] = |\{s_{j,i}, j = 1, ..., i : d(s_j, i) = k\}|$ (for all $k > |O_i|$, we assume $O_i[k] = 0$), similarly $Z_i[k]$ stores the number of substrings ending at i that has d value equals to $-k$ ie. $Z_i[k] = |\{s_{j,i}, j = 1, ..., i : d(s_j, i) = -k\}|$ (for all $k > |Z_i|$, we assume $Z_i[k] = 0$). Notice $\sum_{k=1}^{|O_i|} O_i[k] = T_{i,1}$ and $\sum_{k=1}^{|Z_i|} Z_i[k] = T_{i,3}$.

Suppose we're given O_i, Z_i. Then $T_{i+1,2} = Z_i[1]$ and $T_{i+1,3} = \sum_{k=2}^{|Z_i|} Z_i[k] = T_{i,3} - Z_i[1]$. It remains to show how to compute O_{i+1}, Z_{i+1} in $O(1)$ time. Notice for $k > 1$, we have $O_{i+1}[k] = O_i[k-1]$ and for all $k \geq 1$ we have $Z_{i+1}[k] = Z_i[k+1]$. Also $O_{i+1}[1] = T_{i,2}+1$. It follows $O_{i+1} = O_i.pushfront(T_{i,2}+1)$ and $Z_{i+1} = Z_i.popfront()$.

For the other case where $s_{i+1} = 0$, one can argue similar to above that $P_{i+1} = P_i + T_{i+1,3} + T_{i+1,2} + 1$, $T_{i+1,3} = T_{i,3} + T_{i,2} + 1$, $T_{i+1,2} = O_i[1]$, $T_{i+1,1} = \sum_{k=2}^{|O_i|} O_i[k] = T_{i,1} - O_i[1]$, $Z_{i+1} = Z_i.pushfront(T_{i,2} + 1)$ and $O_{i+1} = O_i.popfront()$.

2.2 algorithm

Below is an $O(n)$ time algorithm running what was described above in iterations.

```
f(s,n){
        T[3] = {0,0,0}
        O,Z be empty lists
        P[n] array of size n

        //base cases
        P[1] = 1
        if(s[1] == 1){
                O.pushfront(1)
                T[1] = 1
        }else{
                Z.pushfront(1)
                T[3] = 1
        }

        //iteration
        for(i = 2; i <= n; ++i){
                if(s[i] == 1){
                        P[i] = P[i-1] + T[1] + T[2] + 1
                        T[1] += T[2] + 1
                        O.pushfront(T[2] + 1) //run this line before the line below to use old T[2]
                        T[2] = Z.front()
                        T[3] -= Z.front()
                        Z.popfront()
                }else{
                        P[i] = P[i-1] + T[3] + T[2] + 1
                        T[3] += T[2] + 1
                        Z.pushfront(T[2] + 1) //run this line before the line below to use old T[2]
                        T[2] = O.front()
                        T[1] -= O.front()
                        O.pushfront()
                }
        }

        return sum(P[1],...,P[n])
```

}