* فرض کنید دو تا لینکد لیست ساده داریم که طول اولی m و دومیn باشد و دقیقا نمیدانیم که n > mاست یا برعکس. این دو تا لینکد لیست از یک نود خاص به بعد کاملا دارای نود های مشترک می شوند.

از شما می‌خواهیم الگوریتمی ارائه دهید که در اردر زمانی O(m + n)این نود خاص که اولین اشتراک این دو لینکد لیست هست را پیدا کند.

خیلی ساده ابتدا دو لینک لیست را گرفته و ترتیب آن ها را بر عکس می کنیم ( صدا زدن reverse operation روی لینک لیست ) ، سپس روی لیست ها به ترتیب پیمایش می کنیم ( همزمان ) تا به دو نود نابرابر برسیم.

شبه کد الگوریتم :

LinkedList A[n];*// A linked list with size n*

LinkedList B[m];*// A linked list with size m*

reverse(A); >> Time = O(n)

reverse(B); >> Time = O(m)

a <- head(A); >> C1

b <- head(B); >> C2

*while* (a != nullptr && b != nullptr) >> Time = O( min(n,m) )

{

*if* (a->next != nullptr && b->next != nullptr &&

a->next->key == b->next->key) >> Time = C3

    {

        a = a->next;

        b = b->next;

    } *else*

    {

*break*;

    }

}

*if* (a->key == b->key) >> Time = C4

{

*return* a;

} *else*

{

*return* nullptr;

}

!\* شرط آخر در صورتی برقرار نیست که دو لیست هیچ اشتراکی نداشته باشند و null pointer باز می گردانیم.

محاسبه پیچیدگی زمانی :

حلقه ی while تا زمانی انجام می شود که به دو نود نابرابر برسد یا حداقل یکی از لینک لیست ها تمام شود. پس پیچیدگی زمانی آن در بدترین حالت حالتی می شود که تمام نود های یک لینک لیست در دیگری باشد که تعداد دفعات برابر اندازه آن لینک لیست است و چون نمی دانیم کدوم لینک لیست سایز کمتری دارد ، پیچیدگی زمانی O( min(n,m)) را برای آن در نظر میگیریم.

در هر خط شبه کد پیچیدگی زمانی آن خط نوشته شده ، توجه کنید که کل عملیات داخل while یک زمان constant است چون به m و n بستگی ندارد. همچنین شرط آخر الگوریتم نیز یک زمان ثابت دارد. پس :

T(A) = O(n) + O(m) + C1 + C2 + min(n,m) + [ min(n,m) – 1] C3 + C4

T(A) = O(n) + O(m) + O( min(m,n) ) + C <= O(n) + O(m) + O(m + n) <= 2 \* O(m + n) = O(m + n)

مثال :

Input = Link A = 1 | 2 | 3 | 4 | 5

Link B = 12 | 20 | 7 | 16 | 3 | 4 | 5

Output = 3

Start:

Reverse(A) >> 5 | 4 | 3 | 2 | 1

Reverse(B) >> 5 | 4 | 3 | 16 | 7 | 20 | 12

In loop :

>> 5 == 5 >> 4 == 4 >> 3 == 3 >> 2 != 16 >> break

>> 3 == 3 >> return 3

امیرحسین نجفی زاده 9831065