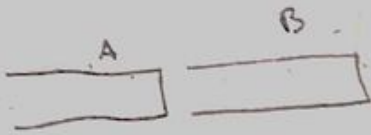


① برای enqueue عناصر را در انتهای (B) push می‌کنیم. (A) زنجیره می‌شود.



Enqueue (A, B, x) {

if (B.full != true) {

B.Push(x);

else

A.Push(x);

if (A.full == true) {

print "Queue is full" }

برای dequeue عناصر را ابتدا در انتهای (B) push می‌کنیم و سپس

آن عناصر را از بالا به ترتیب Pop می‌کنیم و در انتهای (A) Push می‌کنیم.



در نهایت انتهای (A) را Pop می‌کنیم.

شکل ←

Dequeue (A, B) {

while (B.empty != true) {

A.Push (B.Pop); }

A.Pop;

تعدادی . empty و full . به یا خالی . بزرگ است یا  
true , false مشخص می کند .

زمان اجرای Enqueue از مرتبه  $O(1)$  است چونه ربطی به تعداد عناصر  
موجود نداشته دارد .

زمان اجرای Dequeue از مرتبه  $O(n)$  است چونه باید به تمام عناصر  
استدادم را P.P داشته اید Push کند به تعداد  $n$  جستجو دارد .

② از همان الگوریتم Search > کشیدیم استفاده می کنیم منتها آن را در یک حلقه

for می گذاریم تا اعداد 1 از 1 تا n را دانه دانه چک کرده و با اولین عددی که

در کایونتر به آن اشاره کرده بین بیاید و در خیر این مورد NULL برمی گرداند.

Duplicate search (L, x):  
for (int i = 1; i ≤ n; i++) {

\* x.next حلقه ی پشته

اشاره کرده به پشته بعدی است.

x = L.head;

R = 0;

while (x ≠ NULL && x.next ≠ i) {

x = x.next; }

if (x ≠ NULL) {

print ("darghe halghe ast");

R = 1;

break; } }

if (R == 0)

print ("faghe halghe ast");