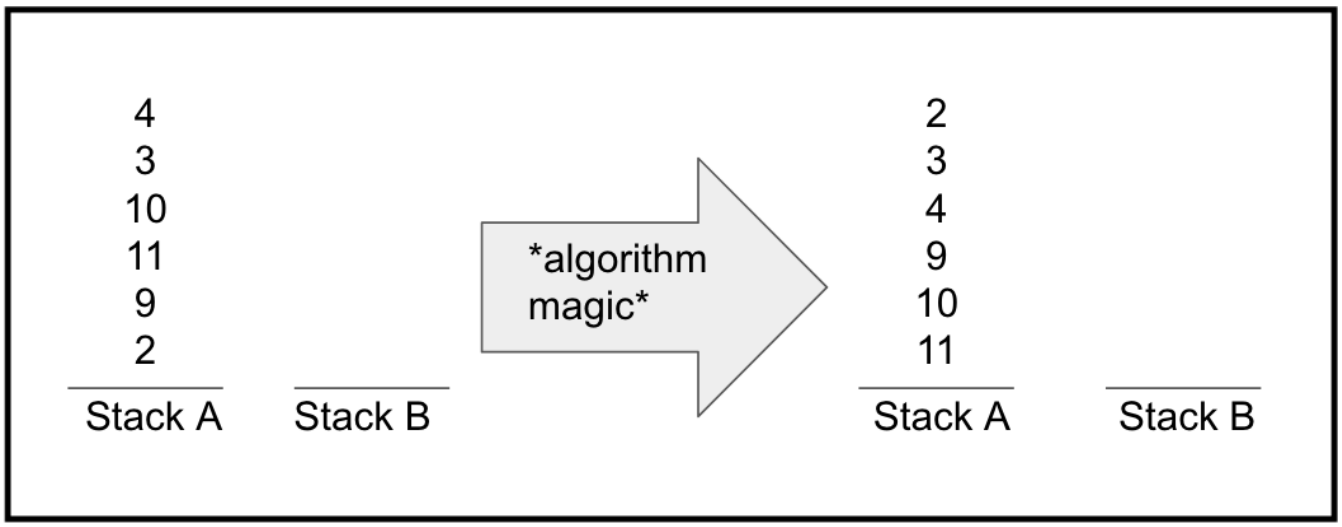
Push_Swap: le moins de mouvements avec deux piles

Jamie Dawson 11 mai 2019 · 7 min de lecture ★

Lorsque je suis devenu étudiant officiel pour la première fois au 42 Silicon Valley , j'ai su très tôt que je voulais m'attaquer à certains des projets d'algorithmes que vous pouvez débloquer.

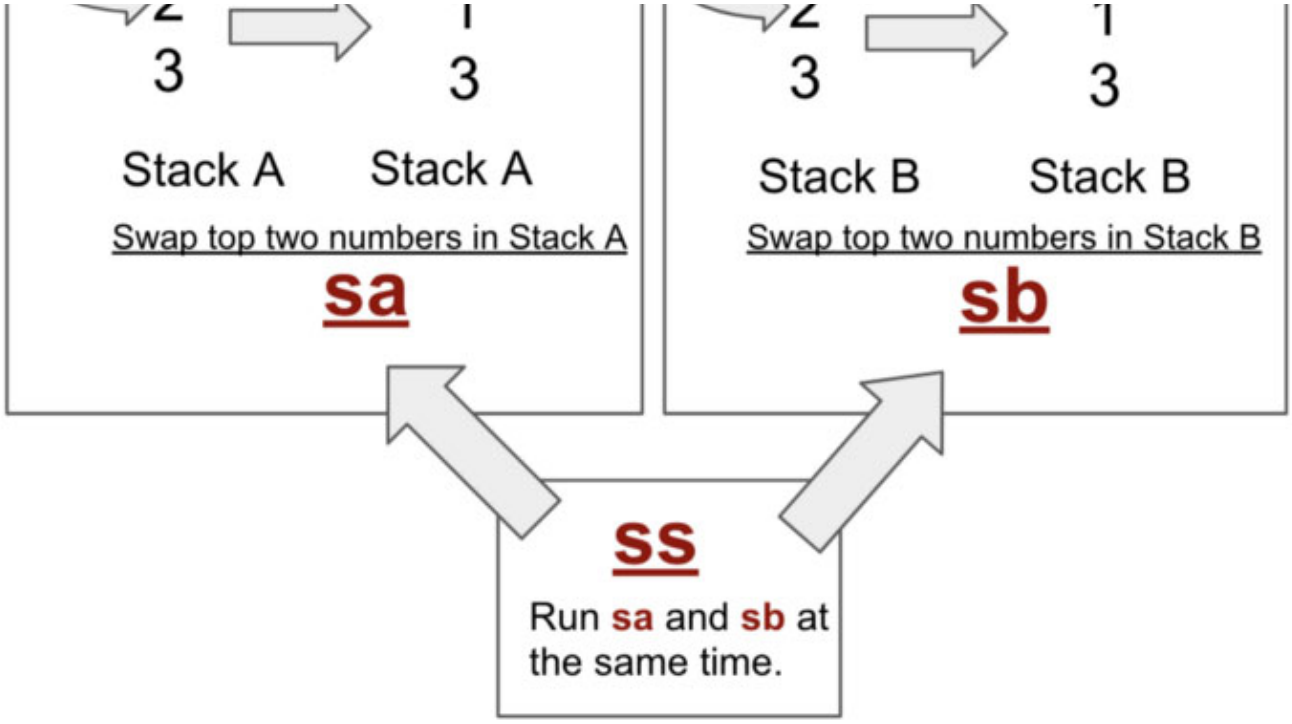
L'un des projets d'algorithme que j'ai débloqué s'appelle **Push_Swap** . L'idée i est simple, vous avez deux piles appelé **Stack A** et **Stack B** . La pile A reçoit une liste aléatoire de nombres non organisés. Vous devez prendre la liste aléatoire des nombres de la pile A et les trier de manière à ce que la pile A soit organisée du plus petit au plus grand. Il n'y a que quelques mouvements que vous êtes autorisé à utiliser pour manipuler les piles que nous allons appeler «Actions». L'objectif principal de ce projet est d'organiser la pile A en aussi peu d'actions que possible.

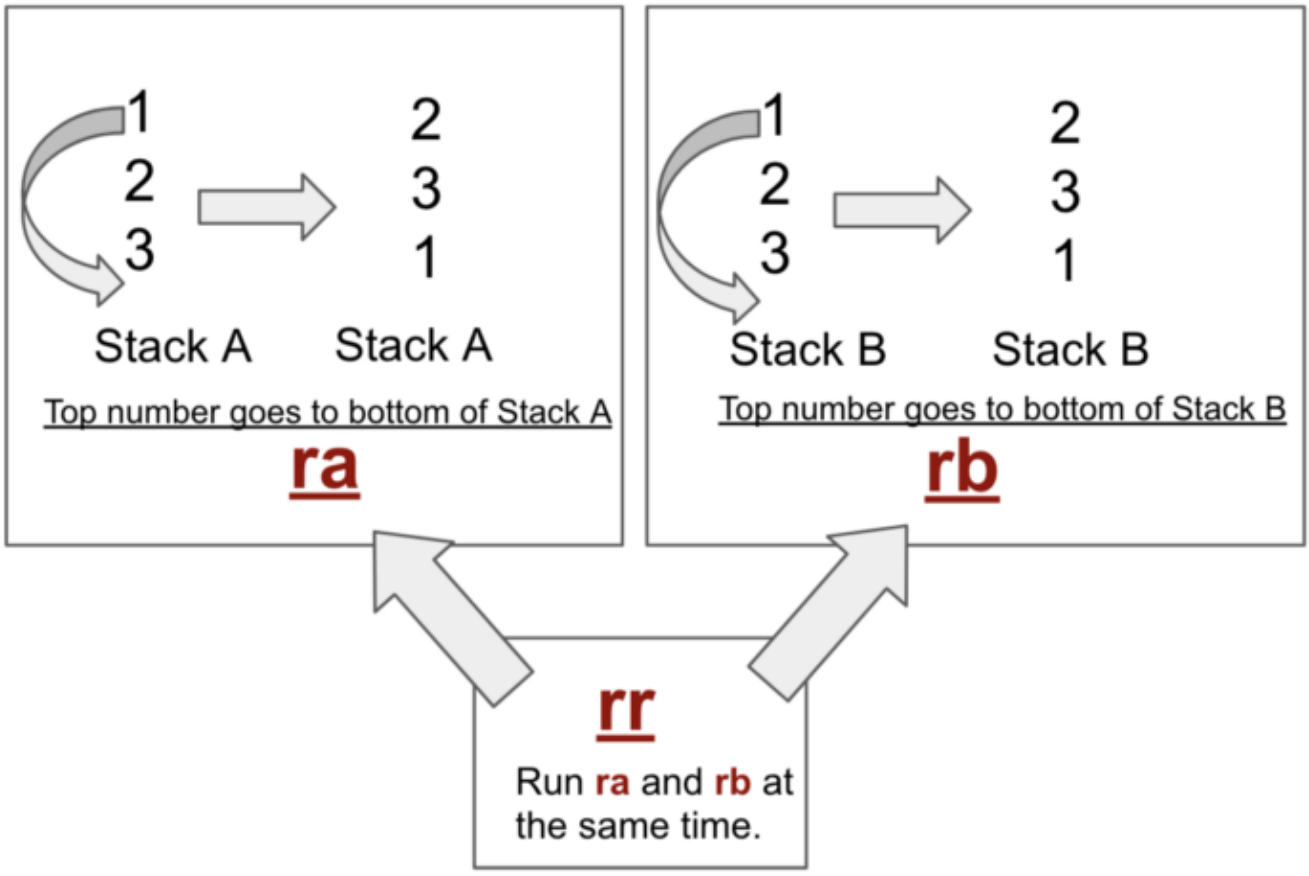


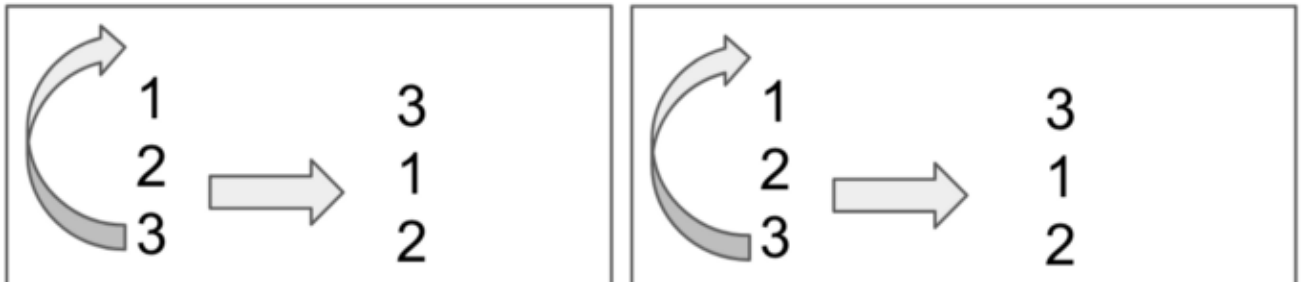
Alors, quelles actions sommes-nous autorisés à utiliser sur les deux piles?

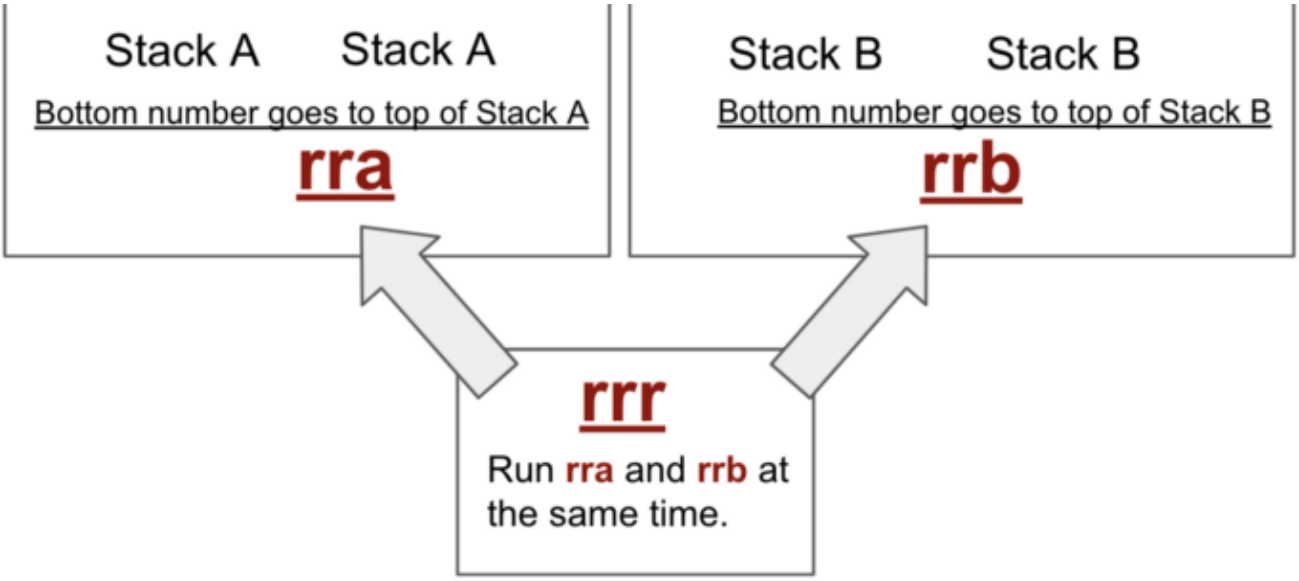
Les actions sont nommées: **sa, sb, ss, ra, rb, rr, rra, rrb, rrr, pa, pb** .

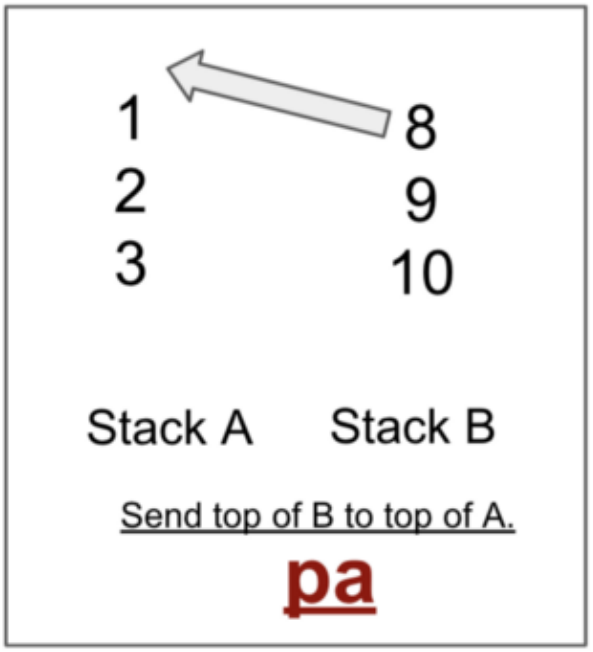
Voici comment ils fonctionnent tous:

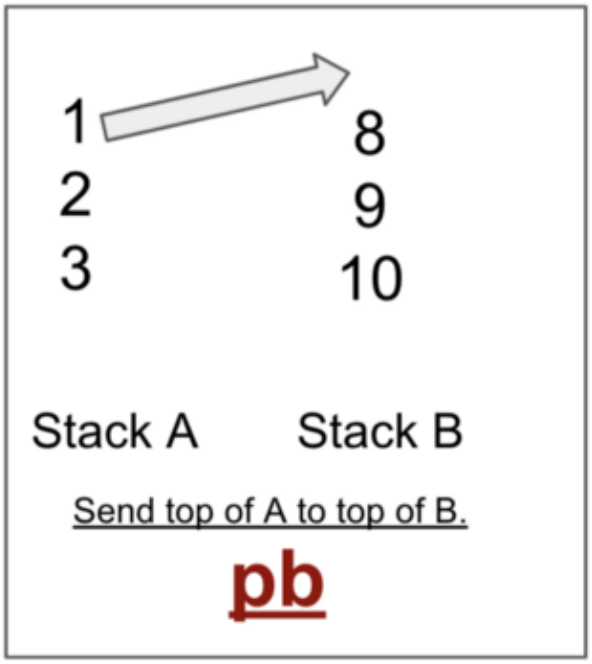

3
3
Stack A
Stack A
Swap top two numbers in Stack A
sa
3
3
Stack B
Stack B
Swap top two numbers in Stack B
sb
ss
Run sa and sb at the same time.


1
2
3
2
3
1
Stack A
Stack A
Top number goes to bottom of Stack A
ra
1
2
3
2
3
1
Stack B
Stack B
Top number goes to bottom of Stack B
rb
rr
Run ra and rb at the same time.


1
2
3
3
1
2
1
2
3
3
1
2





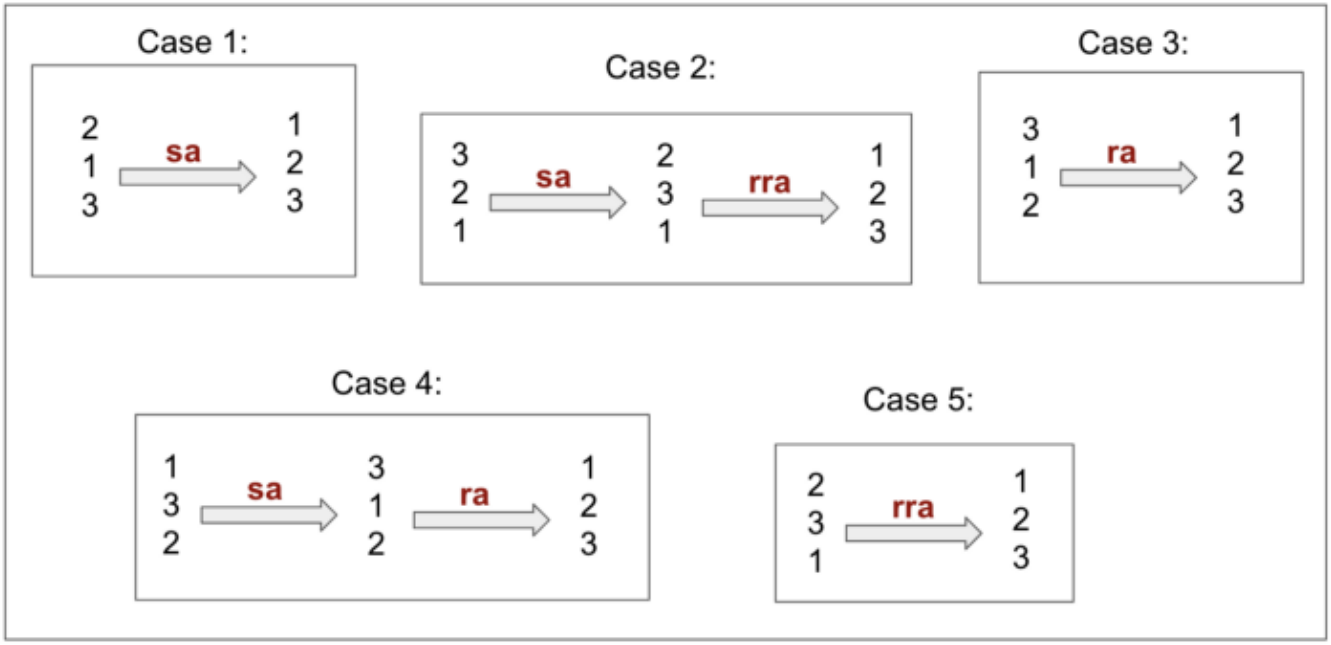


Alors maintenant que nous avons les actions que nous sommes autorisés à utiliser, comment devrions-nous les utiliser? Les algorithmes que nous utiliserons dépendent du nombre de nombres aléatoires appliqués à la pile A. Il y a 4 cas de test principaux que je veux couvrir. Les cas sont **3** , **5** , **100** et **500** . Chaque cas m'oblige à les gérer différemment, je vais donc expliquer comment j'ai optimisé chaque cas un à la fois.

3 nombres aléatoires:

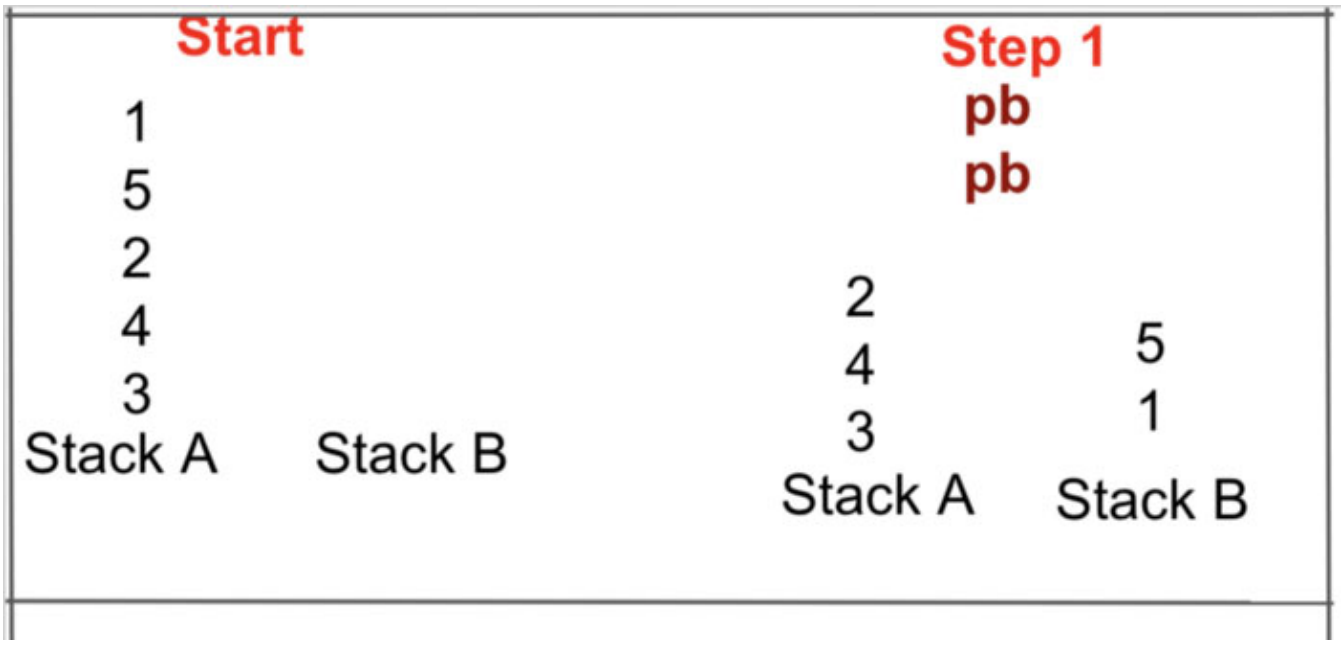
J'ai réalisé qu'il y avait cinq cas possibles pour seulement trois nombres aléatoires mis dans la pile A. Mon objectif est de m'assurer de les trier du plus petit au plus grand en pas plus de deux actions. La façon dont je détermine les actions à utiliser dépend de la position du nombre supérieur, du nombre du milieu et du nombre du bas. Dans chaque

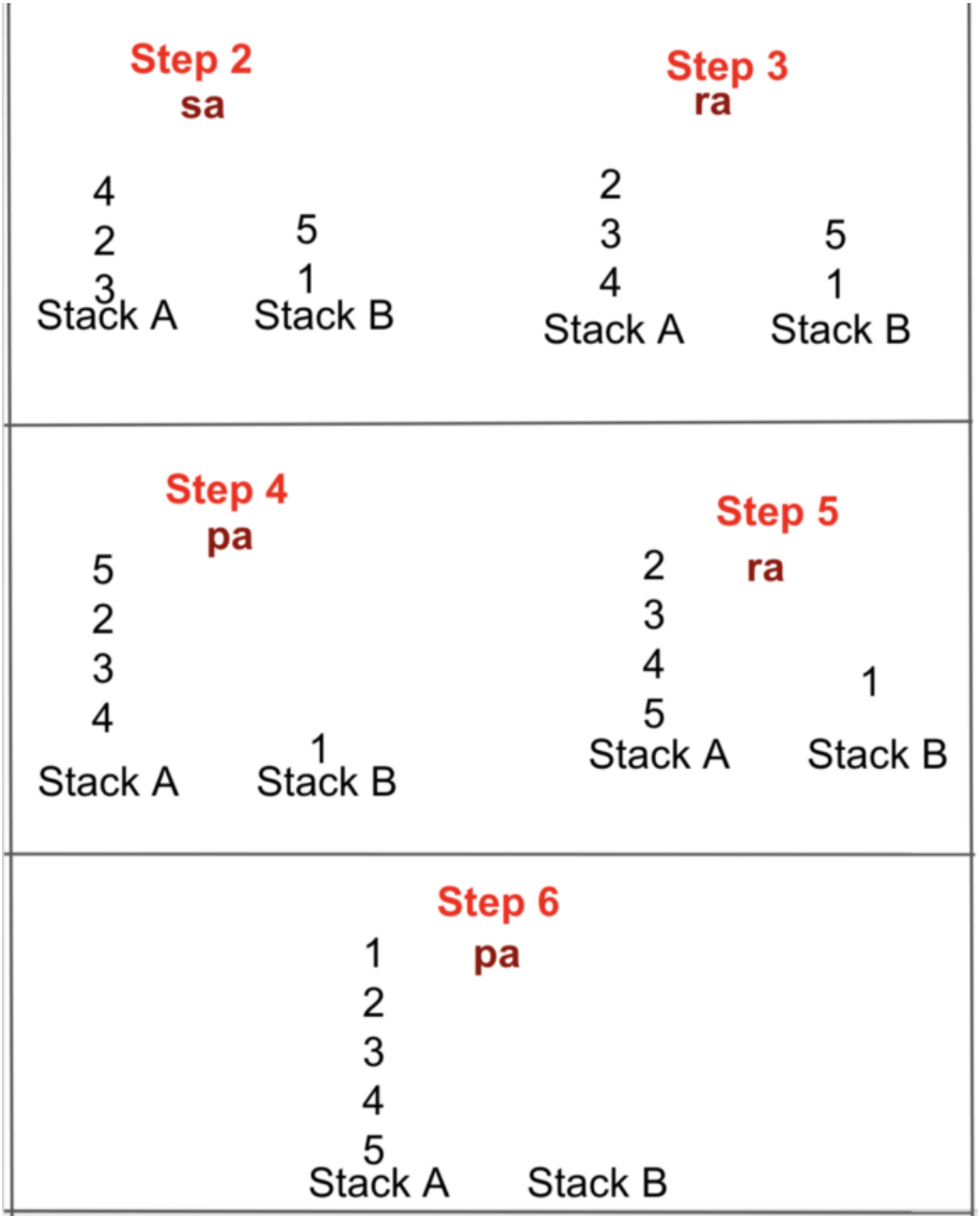
cas, je compare de haut en bas, du milieu en bas et de bas en haut. Selon le nombre plus grand ou plus petit, les actions que j'appelle seront affectées.



5 nombres aléatoires:

Maintenant, nous devons gérer 5 nombres aléatoires placés dans la pile A. Nous ne sommes autorisés que 12 actions. Tout ce qui va au-delà échouera dans cette section du projet. Ce qui est génial à ce sujet, c'est que nous pouvons utiliser la logique de **3 nombres aléatoires** pour optimiser notre code. Tout ce que nous avons à faire est simplement de déplacer les deux premiers nombres du haut de la pile A et de les déplacer vers la pile B. Nous ramènerons ces nombres une fois que les trois nombres de la pile A seront triés du plus petit au plus grand. Je vais appliquer le cas de test **1 5 2 4 3** et montrer comment ma logique fonctionnera.





Étape 1: envoie les deux premiers numéros de la pile A à la pile B.

Étape 2–3: Utilise la logique de **3 nombres aléatoires** pour trier les nombres en A.

Étape 4–6: vérifie que la pile A peut accepter correctement les numéros de la pile B.

Au total, vous utiliserez 7 points d'action. Bien en dessous de notre limite maximale de 12.

100 nombres aléatoires :

Quand j'ai commencé à gérer 100 nombres aléatoires, j'ai pensé que je pourrais simplement utiliser l' algorithme de tri par insertion pour terminer le projet. Les étapes étaient simples:

1. Trouvez le plus petit nombre dans la pile A.-
2. Déplacez le plus petit nombre trouvé vers le haut de la pile A.
3. Poussez ce nombre sur la pile B.
4. Répétez les étapes 1 à 3 jusqu'à ce que la pile A soit vide.
5. Repoussez tout dans la pile A une fois que la pile B a tous les nombres du plus grand au plus petit.

En envoyant le plus petit numéro un à la fois de la pile A à la pile B. j'ai pu faire en sorte que la pile B contienne tous les nombres du plus grand au plus petit. Ensuite, lorsque j'ai renvoyé tous les numéros à la pile A. Ils seront triés du plus petit au plus grand.

Cette approche a *travaillé sur le plan technique* ... Je veux dire, il *est* un algorithme et il a *fait* de trier tous les chiffres ... Mais il était loin d'être suffisamment optimisé pour passer ce projet.

Afin d'optimiser mon idée, j'ai réalisé que je triais 1 gros morceau de 100 numéros. Je pourrais réduire mes actions en triant 5 petits morceaux de 20.

Disons que j'ai une liste aléatoire de 100 nombres de 0 à 99.

Le bloc 1 est compris entre 0 et 19

Le bloc 2 est compris entre 20 et 39

Le bloc 3 est de 40 à 59

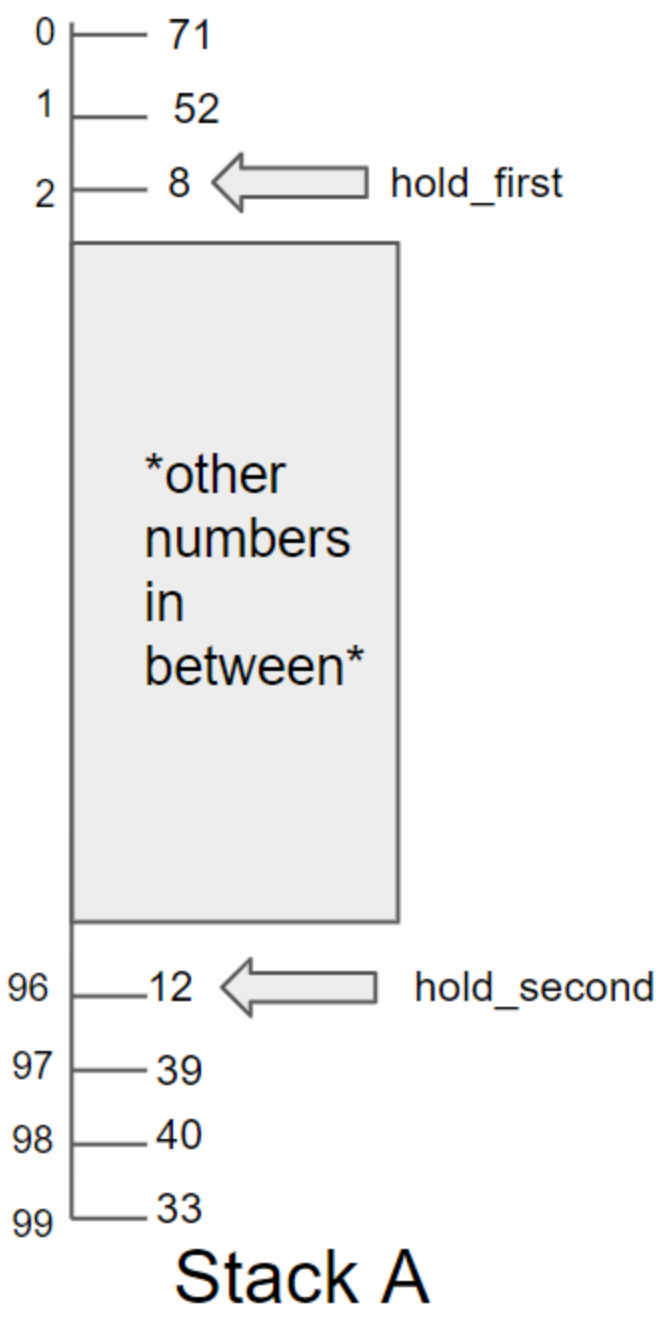
Le bloc 4 est compris entre 60 et 79

Le bloc 5 est compris entre 80 et 99

Étape 1 : Scannez la pile A à partir du haut pour confirmer si l'un des numéros du bloc 1 existe à l'intérieur. Appelons ce numéro **hold_first** .

Étape 2 : Scannez à nouveau la pile A à partir du bas et voyez si un numéro différent du bloc 1 existe dans cette liste. Je vais appeler ce numéro **hold_second** .

Étape 3 : Comparez le nombre de mouvements nécessaires pour obtenir hold_first et hold_second vers le haut.



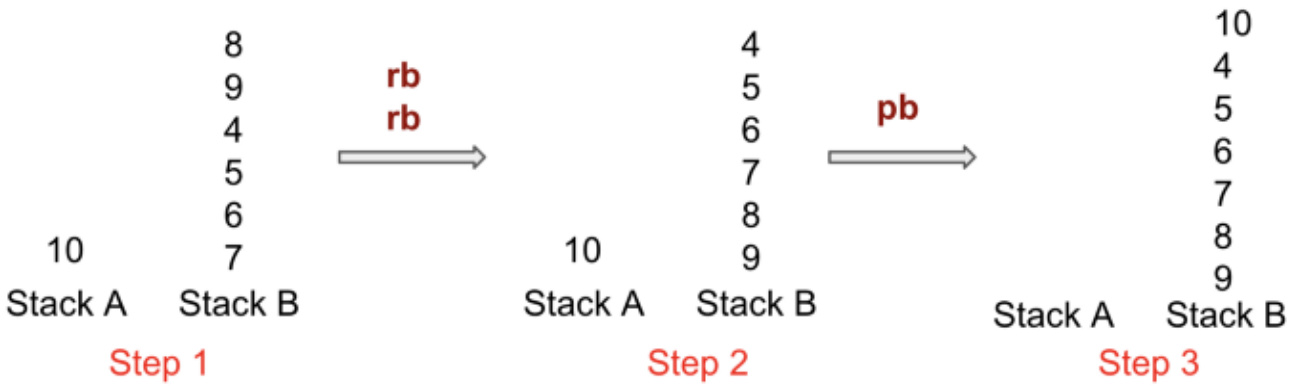
Voici un exemple de deux nombres trouvés dans la pile A. Nous voyons que 8 et 12 sont tous les deux des nombres à l'intérieur du bloc 1 (0–19) et ils sont à l'intérieur de la pile A. Après avoir comparé les deux nombres, nous voyons que pour obtenir 8 en haut de la liste, il faudrait exécuter la commande **ra** 2 fois. Mais pour obtenir le numéro 12 en

haut de la liste, nous aurions besoin d'exécuter la commande **rra** 4 fois. Parce que 8 prend moins de mouvements, nous priorisons 8 et l'envoyons en haut de la pile A.

Avant de passer à l'étape 4, je veux expliquer comment je détermine si ces deux nombres doivent utiliser **ra** ou **rra** . Pour cela, j'utilise juste des mathématiques de base.

Premièrement, je prends le nombre total de nombres dans la liste qui est de 100. Divisez ce nombre par 2, ce qui nous donne 50, ce qui représenterait le milieu de la liste. Ensuite, je trouve l'endroit où se trouve ce numéro dans la liste. Le numéro 8 est à la deuxième place (le numéro du haut étant au point zéro, car il faut zéro mouvement pour amener le numéro du haut en haut de la liste). Je prends donc l'emplacement de 8 et j'essaie de déterminer si l'emplacement du nombre est supérieur ou inférieur à 50. Parce que 2 est inférieur à 50, je sais qu'il serait plus rapide d'exécuter ra sur le numéro 8 au lieu de rra . Mais le numéro 12 est à la 96e place. Parce que 96 est supérieur à 50, je sais que je dois utiliser rra dessus.

Étape 4 : Donc, le numéro correct se trouve maintenant en haut de la pile A. Mais il y a deux choses que nous devons vérifier avant de pousser le nombre vers la pile B.Vous devez vérifier si le nombre que vous poussez est soit plus grand. ou plus petit que tous les autres nombres de la pile B. Puisque nous ne poussons pas seulement le plus petit nombre un à la fois. Nous devons nous assurer que nous n'allons pas provoquer une boucle infinie en essayant de trouver l'endroit parfait pour insérer ce nombre.



Dans l'exemple ci-dessus, le moyen le plus rapide d'obtenir le numéro 10 au bon endroit est de s'assurer que nous obtenons le plus petit nombre (4) et le plaçons sur le dessus de la pile B.Après cela, nous pouvons déplacer le nombre 10 vers le haut de la pile B.

Indépendamment du fait que le nombre en haut de la pile A soit plus grand ou plus petit que tous les nombres de la pile B. Il est toujours bon de vérifier si vous placez le numéro au bon endroit avant de l'envoyer.

Étape 5 : Répétez les étapes 1 à 4 jusqu'à ce que tous les nombres du bloc 1 ne soient plus dans la pile A.

Étape 6 : Répétez les étapes 1 à 4 pour le reste des morceaux afin qu'ils soient traités de la même manière et que toute la pile A se trouve à l'intérieur de la pile B.

Étape 7: Maintenant que la pile A est vide, trouvez le plus grand nombre dans la pile B, déplacez-la vers le haut et poussez-la vers la pile A. Répétez ceci jusqu'à ce que la pile B soit vide. Vous pouvez utiliser la logique de l'étape 3 pour déterminer si vous devez utiliser rb ou rrb pour obtenir rapidement les nombres en haut.

Votre pile A devrait maintenant avoir tous les nombres classés du plus petit au plus grand.

500 nombres aléatoires:

Ce sera donc facile.

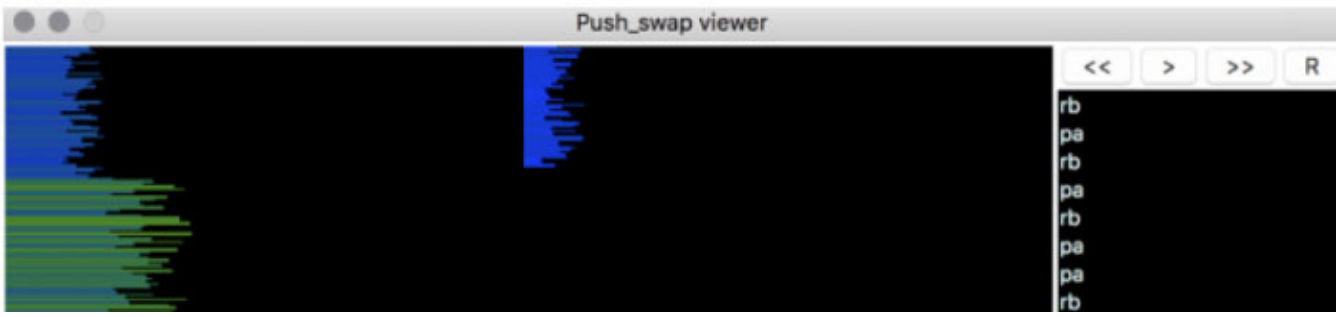
Utilisez simplement la même logique à partir de 100 nombres aléatoires. Mais au lieu de le diviser en 5 morceaux, il suffit de le diviser en 11 morceaux. Pourquoi 11?

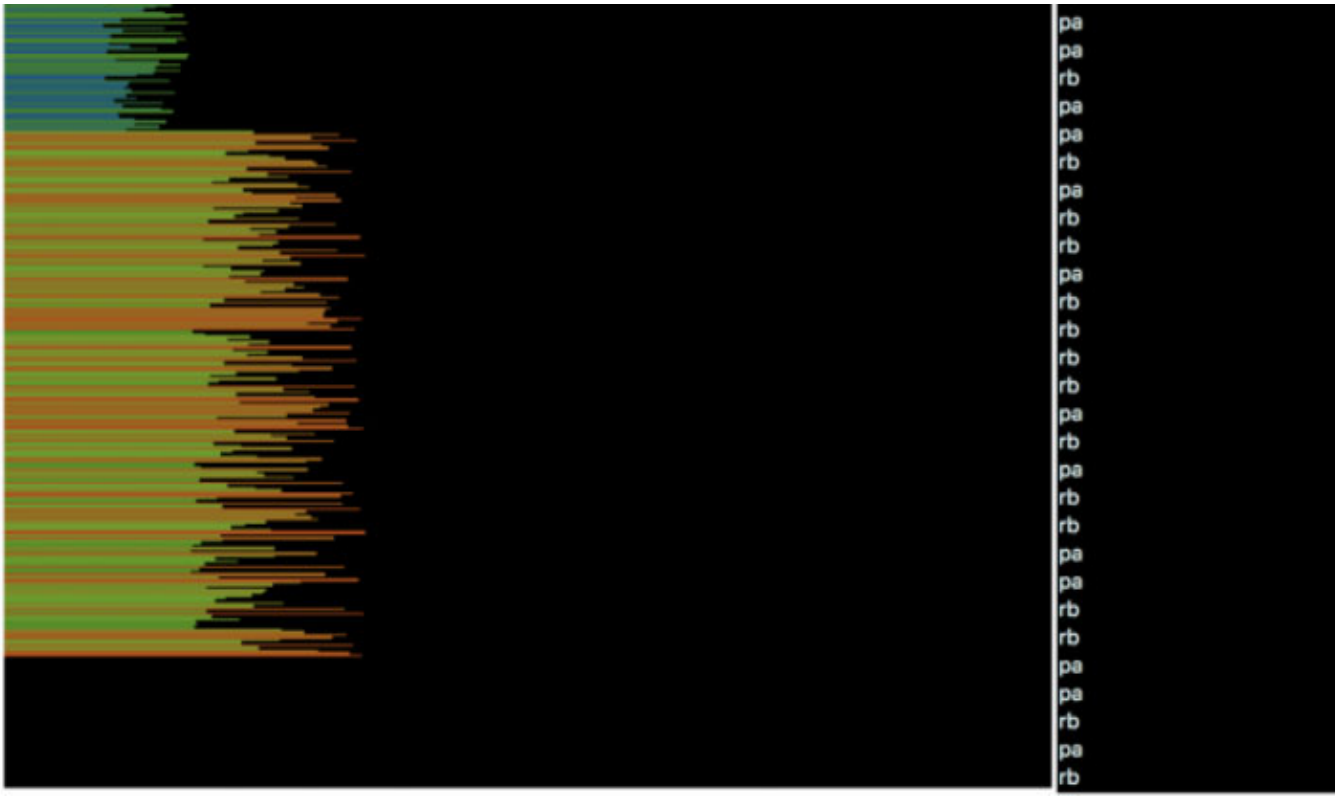
11 morceaux sont ce que j'ai décidé d'utiliser après avoir exécuté plusieurs tests dessus. L'éventail des points d'action que j'ai obtenus était bien inférieur aux autres chiffres sur lesquels je l'ai testé.

Outils que vous pouvez utiliser pour vous aider:

Je ne saurais trop recommander ce visualiseur push_swap. Cet outil a été fait pour push_swap et il m'a vraiment aidé à optimiser mon code.

Cliquez ici pour afficher le compte GitHub du visualiseur .





Programmation | 42 Silicon Valley | Tutoriels de programmation | Tri par insertion | Algorithmes