

Nom : MOSAAD

Prénom : Chehab

Numéro étudiant : 22106126

TP3 : Liste à raccourci - Skiplist.

1) Description de l'archive logicielle fournie :

2) Description du principe de fonctionnement des SkipLists :

3) Travail à réaliser :

1. Définition et construction d'une liste à raccourcis :

1) J'ai créé la SkipList en utilisant 2 structures qui sont les éléments de base de ma SkipList :

1. La structure `s_Node` : J'ai créé cette structure en utilisant 4 champs qui sont :

- Level : le niveau du nœud dans la SkipList.
- Value : la valeur stockée dans le nœud.
- Next : un tableau de pointeurs vers les nœuds suivants dans chaque niveau.
- Prev : un pointeur vers le nœud précédent.

2. La structure `s_SkipList` : J'ai créé cette structure en utilisant 4 champs qui sont :

- Size : le nombre d'éléments stockés dans la SkipList.
- Max_level : le niveau maximal de la SkipList.
- Sentinel : un pointeur vers le nœud sentinelle, qui marque le début de la SkipList.
- Prob : une fonction de distribution de probabilité qui est utilisée pour déterminer le niveau de chaque nœud dans la SkipList.

2) Explication de la fonction `SkipList skiplist_create(int nb_levels)` : La fonction commence par allouer de la mémoire pour la sentinelle de la SkipList. Ensuite, elle alloue de la mémoire pour les pointeurs vers les éléments suivants de la sentinelle, selon le nombre de niveaux souhaité. Ensuite, elle initialise tous les pointeurs suivants de la sentinelle à la sentinelle elle-même, car il n'y a pas encore d'autres éléments dans la SkipList. Elle initialise également le pointeur précédent de la sentinelle à la sentinelle elle-même. Ensuite, la fonction alloue de la mémoire pour la nouvelle SkipList. Elle initialise le niveau de la sentinelle au nombre de niveaux spécifié, et initialise la sentinelle de la SkipList avec la sentinelle que nous avons créée. La fonction initialise également la taille de la SkipList à 0 car il n'y a pas encore d'éléments dans la SkipList, et initialise le niveau maximum de la SkipList au nombre de niveaux spécifié. Enfin, la fonction initialise le générateur de nombres aléatoires utilisé pour générer les niveaux de chaque élément de la SkipList en lui fournissant une graine de 0 (`rng_initialize(0)`) et retourne un pointeur vers la nouvelle SkipList créée.

Explication de la fonction `void skiplist_delete(SkipList d)` : Pour chaque élément, la fonction libère d'abord la mémoire allouée pour les pointeurs `next` de l'élément, puis libère la mémoire allouée pour l'élément lui-même. Une fois que tous les éléments ont été supprimés, la fonction libère la mémoire allouée pour les pointeurs `next` de la sentinelle et pour la sentinelle elle-même. Enfin, la fonction libère la mémoire allouée pour la SkipList.

3) Explication de la fonction `unsigned int skiplist_size(SkipList d)` : La fonction commence par vérifier si la SkipList passée en paramètre est valide en utilisant la fonction `assert`. Si la condition est fausse, le programme s'arrête et affiche un message d'erreur. Sinon la fonction retourne simplement la taille de la SkipList en accédant au champ `size` de la structure `s_SkipList`. Cette fonction possède une implémentation en $O(1)$.

Explication de la fonction `int skiplist_ith(SkipList d, unsigned int i)` : La fonction commence par vérifier si l'indice spécifié est dans la plage de la SkipList à l'aide de la fonction `assert`. Si l'indice est valide, la fonction initialise un pointeur `e` à la sentinelle de la SkipList, puis parcourt chaque élément de la SkipList jusqu'à l'indice spécifié en déplaçant le pointeur `e` au suivant. Une fois que le pointeur `e` pointe sur l'élément à l'indice spécifié, la fonction retourne la valeur de cet élément. Cette fonction possède une implémentation en $O(n)$.

Explication de la fonction `void skiplist_map(SkipList d, ScanOperator f, void *user_data)` : La fonction commence par initialiser un pointeur `e` à la sentinelle de la SkipList. Ensuite, elle parcourt chaque élément de la SkipList en utilisant une boucle `for` ; à chaque itération, le pointeur `e` est déplacé vers le suivant. Également, la fonction `f` est ensuite appliquée à la valeur de l'élément actuel en utilisant l'argument. Cette fonction possède une implémentation en $O(n)$.

- 4) Explication de la fonction `SkipList skiplist_insert(SkipList d, int value)` : La fonction commence par allouer le nouveau nœud à insérer et l'initialiser avec la valeur donnée, un niveau aléatoire (en utilisant la fonction `rng_get_value()` avec le générateur associé à la liste et le niveau maximal de la SkipList), et des pointeurs suivants initialisés à `NULL`.

Si la SkipList est vide, le nouveau nœud est inséré à la position de la sentinelle, et les pointeurs suivants de la sentinelle sont mis à jour pour pointer vers le nouveau nœud.

Sinon, la SkipList est parcourue jusqu'à trouver l'endroit où le nouveau nœud doit être inséré, en fonction de la valeur de la clé du nouveau nœud.

Si un nœud avec la même valeur existe déjà, le nouveau nœud n'est pas inséré et la SkipList est renvoyée sans modification.

Une fois que l'emplacement d'insertion a été trouvé, les pointeurs précédents et suivants des nœuds concernés sont mis à jour pour insérer le nouveau nœud.

Si le niveau du nouveau nœud est inférieur ou égal au niveau du nœud courant, les pointeurs suivants du nouveau nœud sont mis à jour pour pointer vers les mêmes éléments que le nœud courant, et les pointeurs suivants du nœud courant sont mis à jour pour pointer vers le nouveau nœud.

Si le niveau du nouveau nœud est supérieur au niveau du nœud courant, les pointeurs suivants du nouveau nœud sont d'abord mis à jour pour pointer vers les mêmes éléments que le nœud courant pour les niveaux inférieurs au niveau du nœud courant.

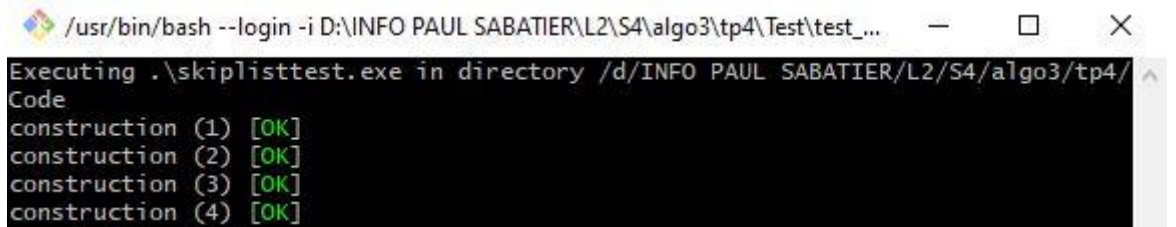
Ensuite, la SkipList est parcourue à partir de l'élément précédent jusqu'à atteindre le niveau du nouveau nœud, et les pointeurs suivants de chaque élément dans la liste sont mis à jour pour pointer vers le nouveau nœud.

Enfin, la taille de la SkipList est incrémentée et la SkipList mise à jour est renvoyée.

- 5) Explication de la fonction `void test_construction(int num)` : la fonction crée une SkipList `d` en utilisant la fonction `buildlist` avec `num` comme argument. Ensuite, elle récupère la taille de la SkipList `d` à l'aide de la fonction `skiplist_size` et l'affiche à l'écran avec un message. Ensuite, la fonction boucle sur tous les éléments de la SkipList à l'aide de la fonction `skiplist_ith` et les affiche à l'écran. Enfin, la fonction supprime la SkipList à l'aide de la fonction `skiplist_delete` et affiche un retour à la ligne.

6)

```
PS D:\INFO PAUL SABATIER\L2\S4\algo3\tp4\Code> .\skiplisttest -c 1
Skiplist (13)
0 1 2 3 4 5 6 7 8 9 11 12 18
PS D:\INFO PAUL SABATIER\L2\S4\algo3\tp4\Code> .\skiplisttest -c 2
Skiplist (13)
0 1 2 3 4 5 6 7 8 9 11 12 18
PS D:\INFO PAUL SABATIER\L2\S4\algo3\tp4\Code> .\skiplisttest -c 3
Skiplist (113)
9 15 23 25 30 35 43 46 48 53 66 79 89 91 93 97 98 109 115 122 127 129 130 141 154 156 160 185 189 190 199 200 208 219 228 229 235 238 248 259 264 265 270 275 276 277 279 281 284 287 290 300 30
9 547 549 552 569 579 588 591 604 606 607 609 610
PS D:\INFO PAUL SABATIER\L2\S4\algo3\tp4\Code> .\skiplisttest -c 4
Skiplist (10922)
0 16 21 22 32 33 38 39 49 56 71 75 84 90 98 103 106 113 114 116 122 128 132 137 144 145 151 158 164 167 168 184 188 191 199 201 204 205 207 212 218 221 223 224 225 228 229 237 248 250 251 252
260 262 266 273 274 277 279 286 289 290 297 301 305 313 314 322 323 327 328 337 345 351 357 362 367 368 380 381 385 386 390 394 395 398 405 407 409 421 426 427 429 441 448 449 450 453 460 466
470 472 475 485 490 492 493 501 504 510 513 515 519 526 535 546 548 551 555 556 567 573 578 582 585 594 598 599 604 610 617 618 620 626 632 633 636 641 646 647 650 653 657 661 662 667 668 672
677 678 683 686 689 691 695 698 701 707 710 714 715 720 722 726 734 737 743 744 746 747 750 751 755 756 766 769 770 778 780 785 786 790 799 800 801 802 805 813 815 816 818 824 825 829 832
845 850 854 856 868 876 880 883 886 887 889 893 894 895 899 902 903 908 911 915 916 918 922 923 924 934 935 936 938 939 944 945 946 951 955 959 973 975 988 989 1000 1005 1010 1017 1018 1019 10
32 1034 1035 1039 1040 1041 1045 1050 1051 1063 1066 1068 1072 1082 1083 1098 1102 1103 1108 1114 1120 1127 1135 1136 1141 1146 1147 1149 1152 1155 1157 1165 1167 1168 1171 1176 1177 1178 1179
1181 1186 1189 1192 1194 1195 1200 1207 1209 1217 1223 1224 1225 1228 1233 1234 1239 1240 1241 1249 1259 1265 1269 1272 1275 1276 1278 1281 1285 1305 1306 1310 1316 1318 1324 1335 1338 1349 1
357 1359 1361 1362 1364 1366 1397 1399 1391 1393 1398 1405 1406 1413 1416 1417 1422 1427 1432 1440 1451 1453 1454 1457 1465 1472 1480 1491 1494 1499 1502 1507 1509 1512 1513 1510 1525 1526 152
7 1529 1530 1541 1543 1555 1556 1558 1560 1561 1563 1565 1568 1570 1574 1577 1582 1583 1585 1591 1599 1601 1603 1615 1620 1631 1630 1645 1650 1660 1662 1665 1666 1672 1683 1693 1695 1700 1716
1734 1738 1743 1745 1749 1754 1762 1768 1773 1779 1788 1794 1795 1796 1801 1803 1808 1819 1820 1822 1828 1834 1836 1843 1847 1848 1852 1854 1855 1857 1865 1873 1877 1881 1889 1890 1891 1892 18
94 1901 1905 1906 1909 1914 1923 1924 1925 1928 1929 1933 1935 1940 1951 1953 1956 1958 1964 1969 1972 1979 1983 1985 1989 1991 1994 2004 2008 2012 2020 2024 2027 2032 2046 2051 2052 2059 2063
2067 2068 2078 2083 2086 2094 2095 2098 2109 2111 2112 2113 2115 2123 2128 2133 2134 2137 2147 2148 2149 2155 2158 2159 2166 2167 2169 2174 2175 2176 2183 2189 2191 2197 2200 2203 2207 2208 2
213 2214 2218 2220 2235 2237 2239 2240 2247 2250 2251 2255 2258 2260 2262 2263 2265 2267 2273 2277 2280 2294 2295 2304 2317 2323 2328 2339 2340 2342 2346 2347 2350 2351 2353 2359
6 2369 2373 2374 2375 2376 2383 2387 2392 2394 2398 2405 2406 2412 2432 2444 2455 2461 2467 2472 2476 2477 2480 2484 2487 2488 2489 2494 2496 2499 2501 2503 2513 2514 2532 2533 2534 2536 2539
2541 2542 2546 2562 2565 2569 2572 2585 2594 2595 2597 2598 2600 2604 2606 2611 2613 2614 2617 2619 2625 2627 2630 2635 2636 2638 2642 2646 2651 2653 2662 2664 2673 2676 2678 2680 2682 2690 26
25 2697 2702 2704 2705 2721 2730 2733 2741 2742 2743 2746 2752 2756 2764 2765 2767 2769 2770 2782 2784 2786 2791 2794 2795 2826 2827 2831 2836 2838 2843 2850 2851 2859 2860 2866 2867 2875 2878
2886 2892 2895 2898 2907 2909 2922 2928 2943 2948 2955 2969 2970 2981 2991 2994 2998 3003 3014 3018 3027 3029 3032 3033 3038 3039 3044 3051 3056 3060 3074 3077 3080 3096 3094 3095 3098 3
100 3104 3106 3115 3116 3117 3118 3127 3145 3149 3153 3162 3165 3167 3168 3169 3173 3174 3175 3179 3181 3186 3207 3209 3216 3230 3240 3242 3246 3250 3257 3262 3264 3271 3273 3282 3285 3287 330
7 3302 3303 3304 3306 3319 3323 3327 3331 3332 3346 3358 3361 3363 3367 3371 3370 3379 3380 3383 3387 3390 3395 3401 3404 3407 3408 3416 3419 3426 3436 3441 3446 3449 3452 3455 3456 3462 3463
3464 3469 3473 3474 3477 3483 3484 3487 3489 3492 3496 3501 3503 3505 3507 3511 3516 3518 3526 3534 3539 3542 3543 3558 3560 3567 3573 3588 3593 3595 3596 3597 3603 3605 3608 3616 3618 3621 36
24 3626 3647 3651 3654 3658 3664 3678 3679 3680 3686 3690 3692 3693 3694 3699 3703 3708 3710 3716 3721 3729 3732 3734 3737 3738 3743 3750 3751 3759 3760 3764 3766 3769 3776 3778 3779 3780 3794
3799 3801 3804 3812 3816 3818 3824 3829 3830 3840 3841 3847 3854 3858 3860 3866 3869 3875 3883 3903 3904 3912 3919 3922 3923 3924 3926 3928 3935 3936 3952 3954 3961 3967 3977 3982 3987 3989 3
990 3997 4001 4008 4011 4017 4026 4040 4052 4059 4078 4084 4087 4090 4097 4099 4104 4105 4125 4127 4131 4135 4137 4147 4148 4153 4154 4155 4157 4167 4170 4174 4182 4183 4186 4188 4199 4200 420
4 4206 4208 4217 4221 4222 4227 4237 4239 4240 4241 4242 4257 4259 4260 4262 4266 4267 4269 4276 4277 4284 4287 4289 4296 4308 4313 4314 4325 4330 4339 4340 4345 4349 4358 4366 4374 4375 4388
4389 4400 4404 4409 4415 4418 4420 4422 4423 4430 4434 4437 4443 4447 4448 4455 4460 4462 4474 4478 4484 4487 4501 4504 4511 4529 4530 4533 4540 4541 4546 4547 4548 4559 4562 4566 4568 4578 45
81 4584 4588 4596 4600 4609 4619 4624 4632 4638 4645 4649 4652 4655 4662 4665 4668 4679 4682 4684 4687 4689 4691 4701 4702 4704 4707 4714 4718 4719 4725 4727 4729 4734 4746 4749 4751 4752 4760
4764 4765 4769 4784 4785 4792 4794 4795 4799 4811 4812 4828 4834 4836 4838 4839 4844 4845 4848 4851 4852 4864 4866 4869 4870 4876 4877 4879 4888 4892 4899 4900 4911 4912 4914 4917 4922 4
923 4924 4929 4931 4933 4934 4941 4944 4945 4946 4948 4952 4953 4954 4956 4957 4960 4967 4973 4976 4979 4981 4985 4989 4995 5010 5011 5020 5027 5029 5030 5036 5042 5051 5057 5061 5065 5075 508
3 5089 5096 5097 5098 5101 5106 5110 5123 5124 5125 5127 5131 5132 5138 5139 5151 5156 5163 5165 5173 5174 5187 5190 5202 5203 5206 5215 5218 5221 5222 5223 5244 5250 5252 5256 5257 5270 5271
5272 5278 5280 5287 5288 5290 5293 5295 5297 5299 5300 5306 5311 5319 5324 5326 5327 5330 5333 5334 5343 5344 5354 5357 5368 5369 5373 5377 5379 5384 5389 5390 5394 5395 5408 5413 5418 5421 54
27 5429 5432 5434 5441 5449 5456 5458 5459 5461 5464 5466 5467 5477 5482 5484 5486 5488 5493 5499 5506 5507 5514 5516 5518 5521 5534 5537 5538 5542 5543 5547 5558 5572 5573 5574 5576 5577 5579
5580 5586 5591 5593 5595 5597 5600 5606 5623 5624 5628 5632 5643 5647 5659 5675 5678 5681 5687 5693 5694 5695 5700 5709 5718 5728 5730 5737 5740 5741 5742 5743 5744 5746 5748 5771 5772 5774 5
775 5785 5790 5800 5801 5808 5810 5813 5815 5816 5823 5826 5835 5836 5843 5844 5845 5847 5849 5853 5855 5858 5859 5861 5865 5869 5872 5874 5875 5880 5883 5884 5885 5888 5889 5892 5894 5899 590
5 5909 5922 5924 5927 5928 5932 5934 5938 5942 5944 5946 5956 5959 5960 5969 5976 5982 5983 5984 5987 5988 5989 5994 5995 5997 6003 6006 6007 6010 6012 6017 6021 6035 6042 6043 6045 6046 6049
6059 6091 6092 6098 6100 6101 6111 6112 6117 6118 6130 6132 6134 6135 6140 6141 6142 6143 6144 6149 6152 6166 6168 6165 6166 6167 6169 6170 6174 6182 6183 6186 6194 6197 6199 6203 6214 6216 62
19 6220 6222 6223 6228 6234 6241 6242 6247 6249 6250 6256 6260 6268 6265 6271 6272 6278 6283 6285 6287 6289 6291 6307 6321 6324 6325 6331 6335 6339 6343 6346 6350 6357 6359 6360 6372 6376 6377 6379 6383
```



2. Recherche d'une valeur dans une liste a raccourcis :

1) Explication de la fonction bool skiplist_search(SkipList d, int value, unsigned int

*nb_operations): La fonction commence par initialiser un pointeur "current" à l'élément sentinelle de la SkipList, qui est le premier élément de chaque niveau. Ensuite, elle parcourt les niveaux de la SkipList, en partant du niveau le plus haut et en descendant vers le niveau 0.

À chaque niveau, la fonction se déplace à l'élément suivant tant que la valeur est inférieure à la valeur cherchée, tout en comptant le nombre d'opérations effectuées avec le pointeur "nb_operations". Si la valeur de l'élément courant est égale à la valeur cherchée, la fonction retourne "true" pour indiquer que la valeur a été trouvée.

Si la fonction arrive au niveau 0 sans trouver la valeur cherchée, elle vérifie l'élément suivant de l'élément courant dans ce niveau. Si la valeur de cet élément est égale à la valeur cherchée, la fonction retourne "true", sinon elle retourne "false" pour indiquer que la valeur n'a pas été trouvée dans la SkipList.

Il y a un erreur dans la fonction est qu'elle ne compte pas les opérations faites si la SkipList n'était pas triée dans le bon ordre mais elle trouve toujours les variables voulues dans le nombre d'opérations correctes.

2) Explication de la fonction void test_search(int num) : la fonction crée une SkipList d en utilisant la fonction buildlist avec num comme argument. Ensuite, elle ouvre un fichier "search[num].txt" en lecture et récupère les valeurs qui y sont stockées. Pour chaque valeur, la fonction effectue une recherche dans la SkipList à l'aide de la fonction skiplist_search et enregistre le nombre d'opérations effectuées dans un tableau tabres. Elle affiche ensuite le résultat de la recherche (true si la valeur est trouvée, false sinon).

Après avoir parcouru toutes les valeurs du fichier, la fonction calcule et affiche des statistiques sur les résultats de la recherche, telles que la taille de la SkipList, le nombre de valeurs trouvées

et non trouvées, ainsi que le nombre minimal, maximal et moyen d'opérations nécessaires pour effectuer les recherches.

3)

```
PS D:\INFO PAUL SABATIER\L2\S4\algo3\tp4\Code> .\skiplisttest -s 1
1 -> true
2 -> true
5 -> true
8 -> true
9 -> true
19 -> false
18 -> true
3 -> true
17 -> false
4 -> true
16 -> false
15 -> false
0 -> true
14 -> false
13 -> false
12 -> true
7 -> true
11 -> true
10 -> false
6 -> true
Statistics :
    Size of the list : 13
Search 20 values :
    Found 13
    Not found 7
    Min number of operations : 1
    Max number of operations : 7
    Mean number of operations : 5
```

```
PS D:\INFO PAUL SABATIER\L2\S4\algo3\tp4\Code> .\skiplisttest -s 2
1 -> true
2 -> true
5 -> true
8 -> true
9 -> true
19 -> false
18 -> true
3 -> true
17 -> false
4 -> true
16 -> false
15 -> false
0 -> true
14 -> false
13 -> false
12 -> true
7 -> true
11 -> true
10 -> false
6 -> true
Statistics :
    Size of the list : 13
Search 20 values :
    Found 13
    Not found 7
    Min number of operations : 1
    Max number of operations : 14
    Mean number of operations : 9
```

```
77 -> false
347 -> true
112 -> false
41 -> false
577 -> false
279 -> true
Statistics :
    Size of the list : 113
Search 369 values :
    Found 77
    Not found 292
    Min number of operations : 1
    Max number of operations : 99
    Mean number of operations : 37
```



```

7711 -> true
34778 -> false
31357 -> true
22548 -> false
1678 -> false
29991 -> false
33853 -> false
33851 -> false
42325 -> false
39097 -> false
15290 -> false
41995 -> false
Statistics :
    Size of the list : 10922
Search 37035 values :
    Found 8209
    Not found 28826
    Min number of operations : 2
    Max number of operations : 9674
    Mean number of operations : 1769

```

```

search (1) [KO]
search (2) [OK]
search (3) [KO]
search (4) [KO]

```

3. Itérateur et recherche linéaire d'une valeur dans une liste :

1) La structure `s_SkipListIterator` : J'ai créé cette structure en utilisant 4 champs qui sont :

1. `current` : un pointeur vers le nœud actuel de l'itérateur.
2. `begin` : un pointeur vers le nœud de début de la liste.
3. `next` : un pointeur de fonction qui prend en entrée un objet de type `SkipListIterator` et renvoie un nouvel objet `SkipListIterator` correspondant au prochain élément de la liste.
4. `d` : une référence à l'objet `SkipList` sur lequel l'itérateur est en train de travailler.

```

void iterate_on_skiplist (SkipList d) {
    SkipListIterator e = skiplist_iterator_create (d, DIRECTION_ITERATOR);
    for ( e = skiplist_iterator_begin (e); ! skiplist_iterator_end (e); e = skiplist_iterator_next (e) ) {
        Do_something_with ( skiplist_iterator_value (e));
    }
}

```

Afin d'exécuter ce code au-dessus, nous devons coder les fonctions `skiplist_iterator_create (d, DIRECTION_ITERATOR)`, `skiplist_iterator_begin (e)`, `skiplist_iterator_end (e)`, `skiplist_iterator_next (e)`, `skiplist_iterator_value (e)`. Également, comme il y a la fonction `skiplist_iterator_create`, nous devons coder la fonction `void skiplist_iterator_delete(SkipListIterator it)` pour pouvoir supprimer l'itérateur et libère la mémoire. Mais aussi, `DIRECTION_ITERATOR` peut prendre une des deux directions `FORWARD`, `BACKWARD` de parcours donc il y a deux autres fonctions qui sont `SkipListIterator forward(SkipListIterator it)` et `SkipListIterator backward(SkipListIterator it)`.

Explication de la fonction `SkipListIterator forward(SkipListIterator it)` : la fonction déplace l'itérateur passé en argument vers le prochain élément en modifiant la valeur de son pointeur « `current` » pour qu'il pointe vers le `next[0]` de l'élément actuel. Enfin, elle retourne l'itérateur mis à jour pour pouvoir être utilisé par le programme appelant.

Explication de la fonction `SkipListIterator backward(SkipListIterator it)` : la fonction déplace l'itérateur passé en argument vers l'élément précédent dans l'ordre croissant de valeur. Pour cela, on modifie le pointeur « `current` » de l'itérateur en le faisant pointer sur le nœud précédent (celui dont la valeur est inférieure) dans la liste, en utilisant le pointeur « `prev` » de l'objet `Node`. Enfin, la fonction retourne l'itérateur mis à jour.

Explication de la fonction `SkipListIterator skiplist_iterator_create(SkipList d, unsigned char w)` : La fonction commence par allouer d'abord de la mémoire pour le nouvel itérateur, puis elle vérifie le sens de l'itérateur en utilisant la variable `w`. Si `w` est `FORWARD_ITERATOR`, l'itérateur est initialisé sur le premier élément de la `SkipList`, sinon, si `w` est `BACKWARD_ITERATOR`, l'itérateur est initialisé sur le dernier élément. La fonction stocke également le premier ou le dernier élément dans la variable `begin` de l'itérateur pour permettre la réinitialisation de l'itérateur plus tard. La fonction configure également la fonction de déplacement de l'itérateur en fonction du sens spécifié (avant ou arrière). Enfin, la fonction stocke la `SkipList` à laquelle l'itérateur est associé et renvoie le nouvel itérateur créé.

Explication de la fonction `void skiplist_iterator_delete(SkipListIterator it)` : la fonction commence par vérifier si le pointeur d'itérateur `it` est non nul avant d'exécuter la fonction `free`, pour éviter une erreur de segmentation. Si `it` est nul, la fonction ne fait rien.

Explication de la fonction `SkipListIterator skiplist_iterator_begin(SkipListIterator it)` : la fonction réinitialise l'itérateur en le positionnant sur le premier élément ou le dernier élément de la `SkipList`, selon le sens de l'itérateur. Elle met à jour le pointeur courant de l'itérateur avec le pointeur vers l'élément initial et retourne l'itérateur mis à jour.

Explication de la fonction `bool skiplist_iterator_end(SkipListIterator it)` : la fonction retourne une valeur booléenne (vrai ou faux) indiquant si l'itérateur a atteint la fin de la `SkipList`, c'est-à-dire si son pointeur courant pointe vers la sentinelle de la `SkipList`.

Explication de la fonction `SkipListIterator skiplist_iterator_next(SkipListIterator it)` : la fonction renvoie l'itérateur suivant en appelant la fonction "next" de l'itérateur sur l'itérateur courant. Le type de l'itérateur renvoyé dépend du sens de l'itération, avant ou arrière.

Explication de la fonction `int skiplist_iterator_value(SkipListIterator it)` : la fonction commence par obtenir la valeur de l'élément courant de l'itérateur. Si l'itérateur n'est pas arrivé à la fin de la `SkipList` (c'est-à-dire si le pointeur courant n'est pas égal à la sentinelle de la `SkipList`), alors la fonction retourne la valeur de l'élément courant. Sinon, si l'itérateur est à la fin de la `SkipList`, la fonction retourne 0.

- 2) Explication de la fonction `void test_search_iterator(int num)` : la fonction lit un fichier contenant des valeurs à chercher dans la `SkipList`. Pour chaque valeur à chercher, la fonction positionne un itérateur au début de la `SkipList` et incrémente un compteur de recherche. Elle parcourt ensuite la `SkipList` à l'aide de l'itérateur, en comparant la valeur cherchée à chaque élément de la liste jusqu'à trouver la valeur cherchée ou atteindre la fin de la liste. Pour chaque recherche, la fonction stocke le résultat (`true` si la valeur a été trouvée, `false` sinon) et le nombre d'opérations effectuées pour trouver la valeur cherchée.

Une fois toutes les recherches effectuées, la fonction calcule des statistiques sur les résultats des recherches, notamment le nombre de fois où la valeur cherchée a été trouvée, le nombre de fois où elle n'a pas été trouvée, le nombre minimum, maximum et moyen d'opérations effectuées pour trouver chaque valeur cherchée. La fonction affiche ensuite les résultats des statistiques et libère la mémoire allouée.

- 3) La fonction `void test_search_iterator(int num)` est meilleure que la fonction `void test_search(int num)` parce qu'elle utilise un itérateur pour parcourir la `SkipList`, ce qui permet de réduire le nombre d'opérations nécessaires pour la recherche.

L'itérateur utilise une boucle `while` pour parcourir la liste, en vérifiant à chaque étape si la valeur cherchée est inférieure à la valeur du nœud suivant. En utilisant un itérateur pour parcourir la `SkipList`, la boucle `while` évite la répétition de certaines opérations inutiles de la boucle `for` dans

skiplist_search, ce qui réduit le temps d'exécution. En outre, la boucle for dans la fonction skiplist_search effectue des opérations de comparaison inutiles dans les niveaux supérieurs de la SkipList, car il est possible que la valeur cherchée soit inférieure à la valeur du nœud suivant dès le niveau supérieur. En utilisant un itérateur, on évite ces opérations inutiles et on réduit encore le temps d'exécution.

Le temps d'exécution est réduit car les itérateurs sont optimisés pour les boucles while, tandis que les boucles for sont moins optimisées. En outre, la boucle for doit effectuer plusieurs opérations à chaque itération, telles que l'accès à la valeur de d->max_level, la comparaison de i à zéro, et la soustraction de un à i. La boucle while dans l'itérateur n'a pas besoin d'effectuer ces opérations à chaque itération, car l'itérateur stocke les informations nécessaires pour parcourir la liste.

4)

```
PS D:\INFO PAUL SABATIER\L2\S4\algo3\tp4\Code> .\skiplisttest -i 1
1 -> true
2 -> true
5 -> true
8 -> true
9 -> true
19 -> false
18 -> true
3 -> true
17 -> false
4 -> true
16 -> false
15 -> false
0 -> true
14 -> false
13 -> false
12 -> true
7 -> true
11 -> true
10 -> false
6 -> true
Statistics :
    Size of the list : 13
Search 20 values :
    Found 13
    Not found 7
    Min number of operations : 1
    Max number of operations : 13
    Mean number of operations : 9
```

```
PS D:\INFO PAUL SABATIER\L2\S4\algo3\tp4\Code> .\skiplisttest -i 2
1 -> true
2 -> true
5 -> true
8 -> true
9 -> true
19 -> false
18 -> true
3 -> true
17 -> false
4 -> true
16 -> false
15 -> false
0 -> true
14 -> false
13 -> false
12 -> true
7 -> true
11 -> true
10 -> false
6 -> true
Statistics :
    Size of the list : 13
Search 20 values :
    Found 13
    Not found 7
    Min number of operations : 1
    Max number of operations : 13
    Mean number of operations : 9
```



```

77 -> false
347 -> true
112 -> false
41 -> false
577 -> false
279 -> true
Statistics :
    Size of the list : 113
Search 369 values :
    Found 77
    Not found 292
    Min number of operations : 1
    Max number of operations : 113
    Mean number of operations : 100

```

```

7711 -> true
34778 -> false
31357 -> true
22548 -> false
1678 -> false
29991 -> false
33853 -> false
33851 -> false
42325 -> false
39097 -> false
15290 -> false
41995 -> false
Statistics :
    Size of the list : 10922
Search 37035 values :
    Found 8209
    Not found 28826
    Min number of operations : 2
    Max number of operations : 10922
    Mean number of operations : 9707

```

```

iterator (1) [OK]
iterator (2) [OK]
iterator (3) [OK]
iterator (4) [OK]

```

4. Suppression d'une valeur dans une liste :

- 1) Explication de la fonction `SkipList skiplist_remove(SkipList d, int value)` : la fonction commence par initialiser un pointeur d'élément "current" à l'élément suivant de la sentinelle de la SkipList, ainsi qu'un pointeur d'élément précédent "prev". Ensuite, elle parcourt la SkipList tant que le pointeur courant n'est pas la sentinelle de la SkipList et que la valeur de l'élément courant n'est pas égale à la valeur donnée. Lorsque l'élément avec la valeur donnée est trouvé, elle stocke le pointeur de l'élément précédent dans "prev". Ensuite, la fonction parcourt les niveaux de l'élément courant en partant de son niveau jusqu'au niveau 0. Pour chaque niveau, elle déplace le pointeur "prev" à l'élément précédent et met à jour les pointeurs suivants de l'élément précédent à l'indice i avec les pointeurs suivants de l'élément courant à l'indice i, pour chaque indice i supérieur ou égal au compteur et inférieur au niveau de l'élément précédent et courant. Après avoir mis à jour les pointeurs suivants des éléments précédents, la fonction met à jour le pointeur précédent de l'élément suivant de l'élément courant avec le pointeur précédent de l'élément courant, puis elle décrémente la taille de la SkipList. Enfin, elle libère la mémoire allouée pour les pointeurs suivants de l'élément courant et pour l'élément courant. Enfin, elle retourne la SkipList modifiée.
- 2) Explication de la fonction `void test_remove(int num)` : la fonction commence par construire une SkipList de taille 'num' en appelant la fonction 'buildlist'. Ensuite, elle ouvre un fichier de test pour les suppressions en appelant 'fopen' et en passant le nom de fichier généré par la fonction 'gettestfilename'. Si le fichier ne peut pas être ouvert, un message d'erreur est affiché et le

programme se termine. La fonction lit ensuite la première ligne du fichier de test, qui contient le nombre de valeurs à supprimer, et exécute une boucle pour supprimer chaque valeur lue dans le fichier de test en appelant la fonction 'skiplist_remove'. Elle ferme ensuite le fichier de test. Après les suppressions, la fonction obtient la taille de la SkipList en appelant 'skiplist_size', puis affiche les éléments de la SkipList, en partant du dernier, en appelant 'skiplist_ith' dans une boucle. Enfin, elle supprime la SkipList en appelant 'skiplist_delete'.

3)

```
PS D:\INFO PAUL SABATIER\L2\S4\algo3\tp4\Code> .\skiplisttest -n 1
Skiplist (6)
18 9 6 4 2 0
PS D:\INFO PAUL SABATIER\L2\S4\algo3\tp4\Code> .\skiplisttest -n 2
Skiplist (6)
9 7 6 5 2 0
PS D:\INFO PAUL SABATIER\L2\S4\algo3\tp4\Code> .\skiplisttest -n 3
Skiplist (39)
610 606 591 588 522 517 475 466 463 460 447 445 428 402 356 338 336 333 313 287 284 276 265 264 259 229 200 199 190 156
154 129 115 93 91 46 43 35 9
PS D:\INFO PAUL SABATIER\L2\S4\algo3\tp4\Code> .\skiplisttest -n 4
Skiplist (561)
49369 49220 49164 49096 49041 48880 48851 48794 48762 48721 48680 48619 48609 48605 48473 48406 48403 48335 48295 48293
48204 48078 47928 47820 47762 47761 47586 47585 47542 47447 47435 47367 47335 47260 47258 47243 47197 47082 47073 47053
47005 46984 46983 46967 46961 46854 46632 46631 46517 46370 46176 46009 46007 45743 45738 45617 45515 45444 45434 45214
44914 44896 44639 44615 44506 44325 44279 44242 44050 43993 43974 43925 43866 43670 43625 43529 43496 43144 43052 42966
42837 42795 42766 42666 42653 42596 42546 42538 42479 42414 42311 42273 42244 42085 41868 41790 41680 41617 41602 41517
41499 41237 41112 41111 41028 41022 41005 40904 40800 40539 40502 40428 40427 40389 40280 40168 40124 40099 39996 39902
39730 39719 39413 38972 38928 38797 38772 38770 38724 38621 38601 38508 38485 38292 38288 38171 38165 37888 37818 37815
37758 37558 37490 37411 37336 37245 37181 37057 37011 36968 36653 36629 36528 36493 36436 36380 36343 36080 36024 35942
35941 35939 35850 35800 35688 35650 35567 35563 35504 35462 35434 35411 35387 35351 35129 35122 35086 34617 34525 34303
34272 34134 34085 34013 33990 33954 33857 33740 33582 33492 33313 33190 33135 33031 32998 32908 32866 32750 32733 32712
32594 32566 32532 32526 32347 32253 32121 32053 31709 31610 31605 31499 31187 31134 30755 30541 30462 30399 30387 30169
29836 29823 29585 29584 29491 29396 29360 29357 29315 29194 29169 29078 29053 29048 29040 28999 28755 28711 28670 28559
28438 28301 28241 28208 27934 27918 27772 27667 27654 27553 27463 27061 27053 27045 27036 27035 26993 26982 26822 26740
26736 26595 26520 26459 26442 26383 26347 26328 26304 26257 26232 26106 26043 25939 25887 25653 25626 25304 25269 25231
25217 25213 25063 25025 24987 24885 24847 24833 24653 24608 24553 24419 24347 24144 24053 23977 23976 23968 23938 23579
23540 23312 23207 22977 22877 22833 22691 22407 22272 22242 22233 22190 22032 21856 21809 21748 21611 21560 21525 21440
21151 21012 20438 20340 20292 20280 20252 20249 20182 20143 20009 19771 19507 19477 19463 19445 19311 19285 19030 18883
18808 18762 18523 18356 18297 18257 18224 18027 17934 17851 17676 17607 17597 17546 17457 17441 17361 17300 17279 17229
17001 16800 16783 16648 16457 16372 16359 16351 16206 16197 16090 16079 16054 15962 15855 15775 15580 15417 15405 15332
15322 15315 15250 15248 15217 15138 15129 14986 14851 14733 14720 14680 14677 14666 14552 14529 13925 13855 13642 13559
13553 13431 13186 13152 13065 13059 13048 12988 12947 12897 12773 12709 12628 12540 12431 12322 12231 11991 11982 11966
11945 11933 11891 11787 11736 11726 11611 11602 11598 11582 11570 11470 11331 11163 10990 10966 10917 10818 10646 10594
10582 10425 10384 10292 10284 10233 10156 10056 10055 10018 9819 9695 9638 9567 9474 9432 9417 9227 9224 9116 8926 8861
8798 8749 8697 8589 8506 8373 8333 8253 8167 7811 7683 7591 7476 7368 7279 7016 7015 6883 6810 6716 6620 6502 6457 6307
6256 6194 6140 6046 5956 5944 5883 5647 5632 5466 5418 5344 5327 5293 5156 5110 4960 4957 4922 4752 4562 4547 4511 4462
4259 4217 4127 3875 3801 3626 3560 3558 3496 3492 3462 3455 3378 3367 3363 3108 3018 2998 2741 2704 2646 2572 2477 2375
2338 2331 2317 2134 2133 2051 1762 1754 1457 1453 1451 1186 1114 1018 944 894 887 818 790 618 610 594 501 323 266 122 11
4
```

```
remove (1) [OK]
remove (2) [OK]
remove (3) [OK]
```

```
/usr/bin/bash --login -i D:\INFO PAUL SABATIER\L2\S4\algo3\tp4\Test\test_...
Executing .\skiplisttest.exe in directory /d/INFO PAUL SABATIER/L2/S4/algo3/tp4/
Code
construction (1) [OK]
construction (2) [OK]
construction (3) [OK]
construction (4) [OK]
search (1) [KO]
search (2) [OK]
search (3) [KO]
search (4) [KO]
iterator (1) [OK]
iterator (2) [OK]
iterator (3) [OK]
iterator (4) [OK]
remove (1) [OK]
remove (2) [OK]
remove (3) [OK]
```