9. Sequence Data II

12. Sorting and Searching

Sorting (1)

```
#include <iostream>
#include <vector>
void selection_sort(std::vector<int>& a) {
   int n = a.size();
   for (int i = 0; i < n - 1; i++) {
      int small = i;
      for (int j = i + 1; j < n; j++)
         if (a[j] < a[small])
            small = j; // Found a smaller value
      if (i != small)
         swap(a[i], a[small]);
   }
}
void swap(int& a, int& b) { // std::swap, <algorithm>
   int temp = a;
   a = b;
   b = temp;
}
```

Sorting (2)

```
void print(const std::vector<int>& a) {
   int n = a.size();
   if (n > 0) {
      for (int i = 0; i < n; i++)
         std::cout << a[i] << ' ';
   }
    std::cout << '\n';
}
int main() {
   std::vector<int> list{23, -3, 4, 215, 0, -3, 2, 23, 100, 88, -10};
   std::cout << "Before: ";
   print(list);

   selection_sort(list);
   std::cout << "After: ";
   print(list);
}
```

Flexible Sorting

```
bool less_than(int a, int b) {
   return a < b;
}
bool greater_than(int a, int b) {
   return a > b;
}
void selection_sort(std::vector<int>& a, bool (*compare)(int, int)) {
   int n = a.size();
   for (int i = 0; i < n - 1; i++) {
      int small = i;
      for (int j = i + 1; j < n; j++)
         if (compare(a[j], a[small]))
            small = j;
      if (i != small)
         std::swap(a[i], a[small]);
   }
}
// selection_sort(list, less_than);
```

Search (1)

```
#include <iostream>
#include <vector>
#include <iomanip>
int locate(const std::vector<int>& a, int seek) {
   int n = a.size();
   for (int i = 0; i < n; i++)
      if (a[i] == seek) return i;
   return -1;
}

void print(const std::vector<int>& v) {
   for (int i : v)
       std::cout << std::setw(4) << i;
   std::cout << std::endl;
}
```

Search (2)

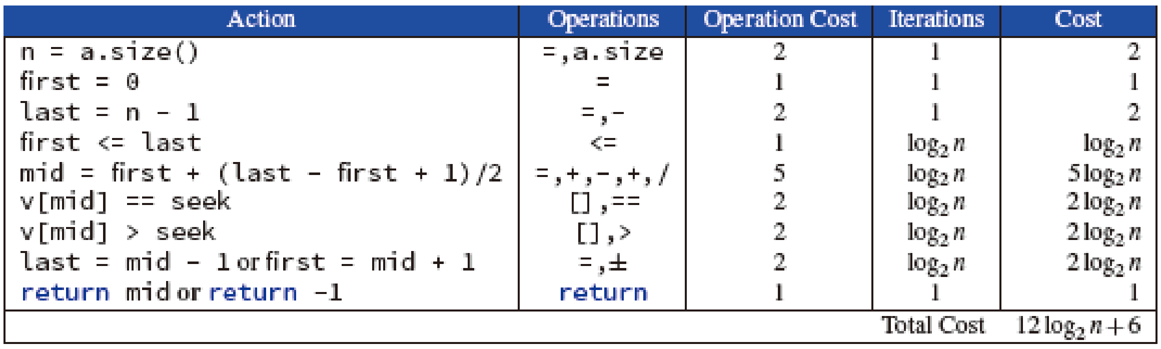
```
void display(const std::vector<int>& a, int value) {
   int position = locate(a, value);
   if (position >= 0) {
      std::cout << value << " in ";
      print(a);
   }
   else {
      std::cout << value << " not in ";
      print(a);
   }
}
int main() {
   std::vector<int> list{ 100, 44, 2, 80, 5, 13, 11, 2, 110 };
   display(list, 13);
   display(list, 2);
   display(list, 7);
   display(list, 100);
}
```

Binary Search (1)

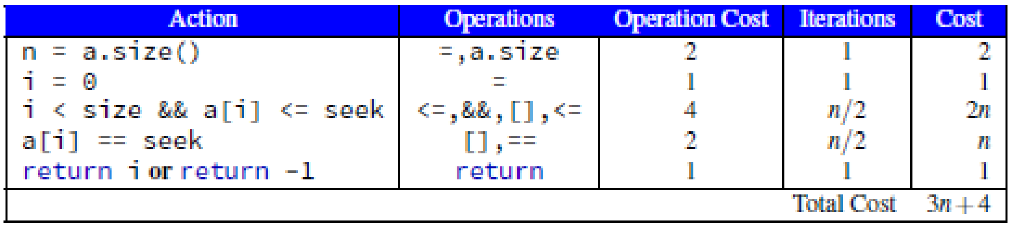
```
int binary_search(const std::vector<int>& a, int seek) {
   int first = 0, last = a.size() - 1, mid;

   while (first <= last) {
      mid = first + (last - first + 1)/2;  // (f+l+1)/2
      if (a[mid] == seek)
         return mid; // Found it
      else if (a[mid] > seek)
         last = mid - 1;
      else
         first = mid + 1;
   }
   return -1;
}
```

Binary Search (2)



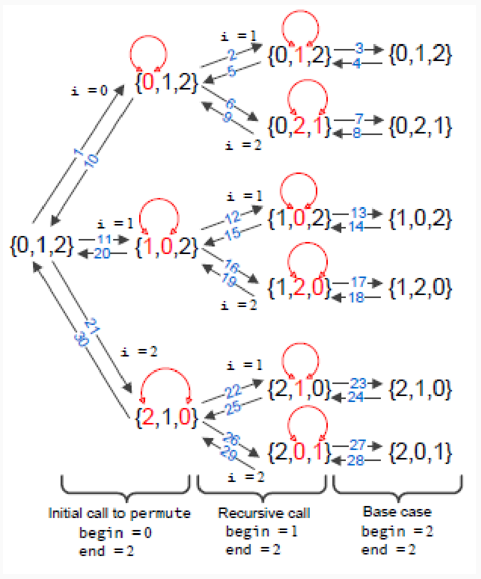
Action	Operations	Operation Cost	Iterations	Cost
`n = a.size()`	`=,a.size`	2	1	2
`first = 0`	`=`	1	1	1
`last = n - 1`	`=,-`	2	1	2
`first <= last`	`<=`	1	$\log_2 n$	$\log_2 n$
`mid = first + (last - first + 1)/2`	`=,+,-,+,/`	5	$\log_2 n$	$5\log_2 n$
`v[mid] == seek`	`[],==`	2	$\log_2 n$	$2\log_2 n$
`v[mid] > seek`	`[],>`	2	$\log_2 n$	$2\log_2 n$
`last = mid - 1` or `first = mid + 1`	`=,±`	2	$\log_2 n$	$2\log_2 n$
`return mid` or `return -1`	`return`	1	1	1
			Total Cost	$12\log_2 n+6$



Action	Operations	Operation Cost	Iterations	Cost
`n = a.size()`	`=,a.size`	2	1	2
`i = 0`	`=`	1	1	1
`i < size && a[i] <= seek`	`<=,&&,[],<=`	4	$n/2$	$2n$
`a[i] == seek`	`[],==`	2	$n/2$	n
`return i` or `return -1`	`return`	1	1	1
			Total Cost	$3n+4$

Vector Permutations (1)

```
void permute(std::vector<int>& a, int begin, int end) {
   if (begin == end) {
      print(a);
      std::cout << '\n';
   }
   else {
      for (int i = begin; i <= end; i++) {
         std::swap(a[begin], a[i]);
         permute(a, begin + 1, end);
         std::swap(a[begin], a[i]);
      }
   }
}
```



Vector Permutations (2)

```
#include <iostream>
#include <vector>
#include <algorithm>
void print(const std::vector<int>& a) {
   // …
}
int main() {
   std::vector<int> nums { 0, 1, 2, 3 };
   std::cout << "--------------\n";
   do {
      print(nums);
      std::cout << '\n';
   } // Compute the next ordering of elements
   while (next_permutation(std::begin(nums), std::end(nums)));
}
```

Randomly Permuting a Vector (1)

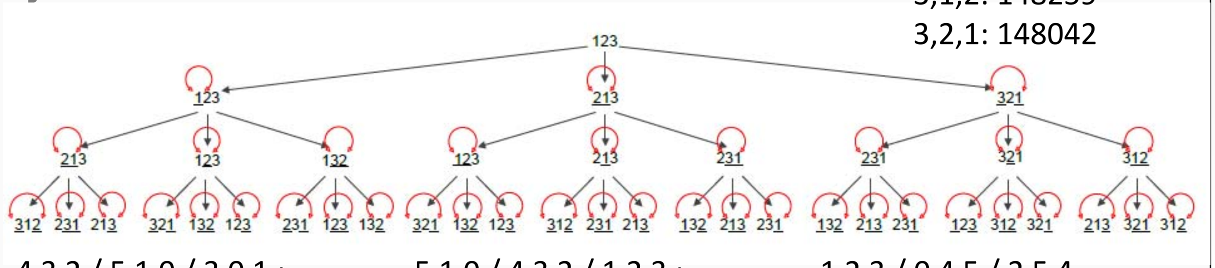
```
int random(int begin, int end) {  // [begin, end)
   if (begin >= end) return 0;
   else {
      int range = end – begin;
      return begin + rand()%range;
   }
}

void permute(std::vector<int>& a) {
   int n = a.size();
   for (int i = 0; i < n - 1; i++) {
      std::swap(a[i], a[random(i, n)]);
   }
}
```

Randomly Permuting a Vector (2)

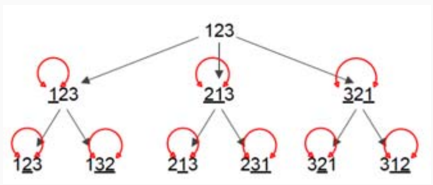
```
void faulty_permute(std::vector<int>& a) {
   int n = a.size();
   for (int i = 0; i < n; i++) {
      std::swap(a[i], a[random(0, n)]);
   }
}
```

```
1,2,3: 148307
1,3,2: 184899
2,1,3: 185359
2,3,1: 185134
3,1,2: 148259
3,2,1: 148042
```



4 3 2 / 5 1 0 / 3 0 1 ;   5 1 0 / 4 3 2 / 1 2 3 ;   1 2 3 / 0 4 5 / 2 5 4

0: 4
1: 5
2: 5
3: 5
4: 4
5: 4


123
123
213
321
123
132
213
231
321
312