Recursion

- recursive form of sequential search

```
SeqSearch ( list head, int goal ){
    if (head is NULL)   return NULL;

    if (head.num is goal)   return head;

    (else)   return SeqSearch (head → next, goal);
}
```

"linear recursion" (one call to itself)

- recursive form of binary search (arr ordered)

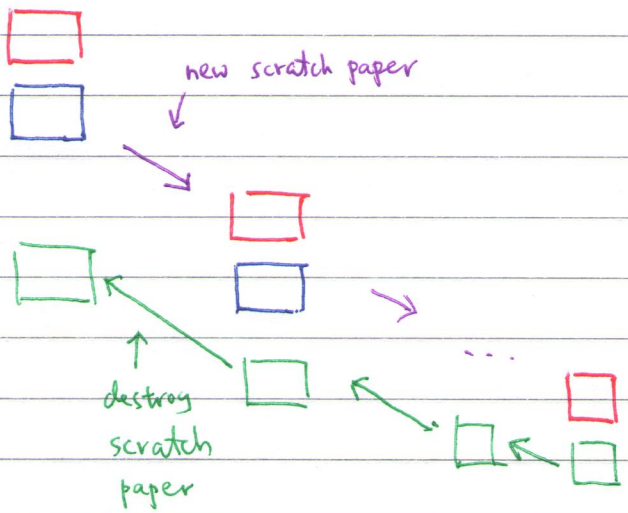
```
BinSearch ( int left, int right, int goal ){
    if ( left > right )   return NONE;

    if ( arr[ (left+right)/2 ] is goal )   return (left+right)/2;
                                  (middle)

    (else) {  if (arr[middle] > goal)
                  return BinSearch( left, middle-1, goal );
              if (arr[middle] < goal)
                  return BinSearch (middle+1, right, goal);
```

Still "linear recursion"

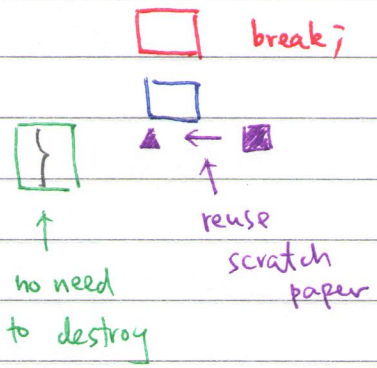
* ① often make program intention clear.
② careful about termination condition
③ at the expense of space (why?)

* tail recursion : "space saving" for special linear recursion

```
void f(▲) {
    if (...) terminate recursion;
    A
    f(■)
}
```




```
while (true) {
    break;
}
```



e.g. Seq Search

```
while (true) {
    if (head is NULL) break;
    if (head.num is goal) break;
    head ← head.next;
}
```

" recursive thinking
non-recursive implementation (if possible) "

* more complicated recursions : reading assignments, will encounter later