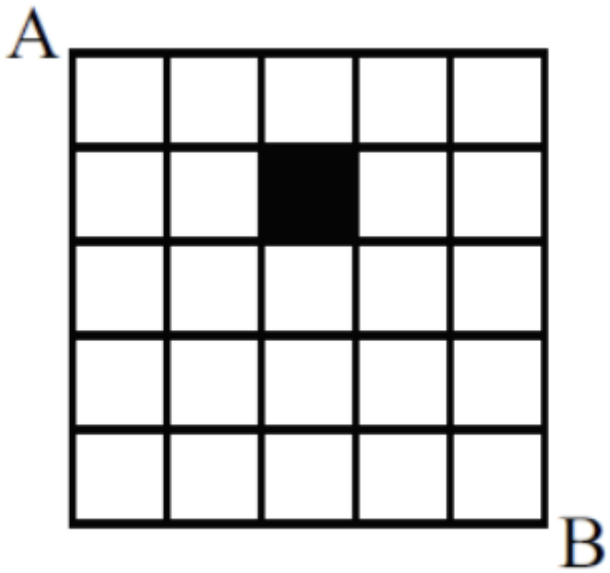
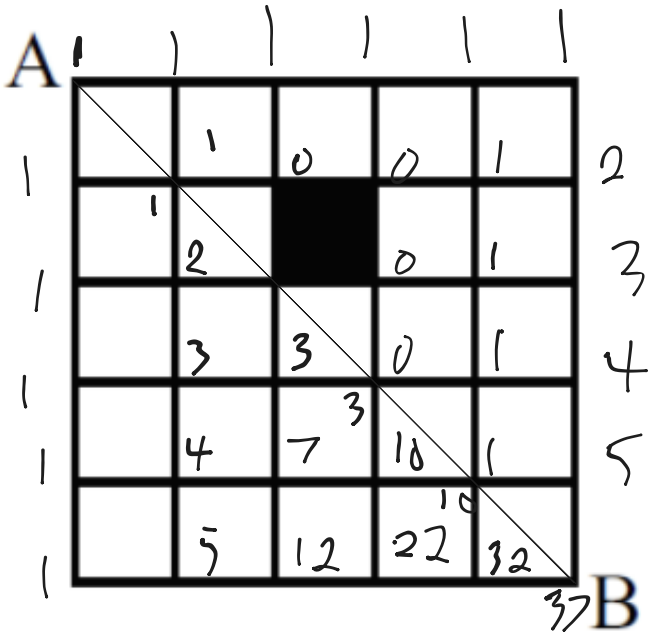
Test 3 Study Guide

1. Given an array A[] of n numbers and a number s, determine in O(n) time if there are two elements in A whose sum is s. (Hint: Hashtable)
    1. Basically, check if the compliment (s - A[i]) is equal to A[j] by checking a hashmap full of the compliments so far. If not, store that compliment (s-A[j]), and go to the next element, check again. Continue unti we either find one (true) or reach the end of the list (false)
    2. TwoSum(A[0...n-1], s)
        1. Hashmap compliments
        2. for i = 0 to n-1
            1. if A[i] in compliments
                1. return true
            2. else
                1. compliments += s - A[i]
        3. return false
2. Bonus points (3 pts): determine in O(n) time if there is a contiguous subarray A[i, … , j] whose sum is s.
    1. Make an array B whose elements are B[i] = A[1] + A[2] + ... + A[i]
    2. B[j] - B[i-1] = $\sum_{k=i}^{j} A[k]$
        1. B[j] = $\sum_{k=0}^{j} A[k]$
        2. B[i-1] = $\sum_{k=0}^{i-1} A[k]$
        3. Just check if this is S, so we dont have to do the whole summation
    3. B[j] - S = B[i-1]
        1. Check if its in the hash table. If so great, if not move to next element
    4. ContSum(A, S)
        1. if empty(A) return false
        2. prefixSum = 0
        3. HashTable prefixSumDict = {0:-1}
        4. //{value:index}
        5. foreach i, num in enumerator(A)
            1. //i is index, num is value at that index, and we iterate over all of A
            2. prefixSum = prefixSum + num
            3. if prefixSum-S in prefixDict

        1. return true
    4. if prefixSum !in prefix_dict
        1. prefixSumDict[prefix_sum] = i
    6. return false
3. In the figure, each segment between two adjacent vertices has a length of 1 unit. How many ways can you go from A to B along a sequence of 10 edge segments without touching a side or vertex of the shaded square and without crossing the line that connects A and B? Solve the problem with a dynamic programming approach.



    1. From A, or any point, we can only go either down or to the right
    2. From each vertex on the top and left edge, the number of paths is 1



3.

4. Describe a Θ(m + n) time algorithm to merge two min heaps (of size m and n, respectively) into a single min heap.
    1. Use bottom up, its linear dumbass
    2. Answer: Create an empty heap of size and copy elements of both heaps to the new heap. Then use a bottom-up process to construct the heap.