countpath

```
public class Main {
  public static void main(String[] args) throws Exception {
    BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
    int n = Integer.parseInt(br.readLine());
    long[] arr = new long[n + 1];
    arr[0] = 1;
    for (int i = 1; i <= n; i++) {
      if (i >= 1) {
        arr[i] += arr[i - 1];
      }
      if (i >= 2) {
        arr[i] += arr[i - 2];
      }
      if (i >= 3) {
        arr[i] += arr[i - 3];
      }
    }
    System.out.println(arr[n]);
  }


}
```

Climb Stairs With Variable Jumps

```
public class Main {
  public static void main(String[] args) throws Exception {
    BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
    int n = Integer.parseInt(br.readLine());
    int[] arr = new int[n];
    for (int i = 0; i < arr.length; i++) {
      arr[i] = Integer.parseInt(br.readLine());
    }

    int[] dp = new int[n + 1];
    dp[n] = 1;

    for (int i = n - 1; i >= 0; i--) {
      if (arr[i] > 0) {
        for (int j = 1; j <= arr[i] && i + j < dp.length; j++) {
          dp[i] += dp[i + j];
        }
      }
    }
    System.out.println(dp[0]);
  }

}
```

Climb Stairs With Minimum Moves

```
public class Main {

  public static void main(String[] args) throws Exception {

    BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

    int n = Integer.parseInt(br.readLine());

    int[] arr = new int[n];

    for (int i = 0; i < arr.length; i++) {

      arr[i] = Integer.parseInt(br.readLine());

    }

    Integer[] dp = new Integer[n + 1];

    dp[n] = 0;

    for (int i = n - 1; i >= 0; i--) {

      if (arr[i] > 0) {

        int min = Integer.MAX_VALUE;

        for (int j = 1; j <= arr[i] && i + j < dp.length; j++) {

          if(dp[i + j] != null){

            min = Math.min(min, dp[i + j]);

          } }

        if(min != Integer.MAX_VALUE){

          dp[i] = min + 1;

        }   }

    }

    System.out.println(dp[0]);

  }}
```

Min Cost In Maze Traversal

```
public class Main {

        public static void main(String[] args) throws Exception {

    int[][] arr = new int[n][m];

            for (int i = 0; i < n; i++) {

               String str = br.readLine();

               for (int j = 0; j < m; j++) {

                  arr[i][j] = Integer.parseInt(str.split(" ")[j]);

               }

            }

            int[][] dp = new int[arr.length][arr[0].length];

            for(int i = arr.length - 1; i >= 0; i--){

               for(int j = arr[0].length - 1; j >= 0; j--){

                  if(i == arr.length - 1 && j == arr[0].length - 1){

                     dp[i][j] = arr[i][j];

                  } else if(i == arr.length - 1){

                     dp[i][j] = arr[i][j] + dp[i][j + 1];

                  } else if(j == arr[0].length - 1){

                     dp[i][j] = arr[i][j] + dp[i + 1][j];

                  } else {

                     dp[i][j] = arr[i][j] + Math.min(dp[i][j + 1], dp[i + 1][j]);

                  }

               }

            }System.out.println(dp[0][0]);}}
```

Goldmine

```
public class Main {

  public static void main(String[] args) throws Exception {

    int[][] dp = new int[arr.length][arr[0].length];

    for (int j = arr[0].length - 1; j >= 0; j--) {

      for (int i = arr.length - 1; i >= 0; i--) {

        if (j == arr[0].length - 1) {

          dp[i][j] = arr[i][j];

        } else if (i == arr.length - 1) {

          dp[i][j] = arr[i][j] + Math.max(dp[i][j + 1], dp[i - 1][j + 1]);

        } else if (i == 0) {

          dp[i][j] = arr[i][j] + Math.max(dp[i][j + 1], dp[i + 1][j + 1]);

        } else {

          dp[i][j] = arr[i][j] + Math.max(dp[i][j + 1], Math.max(dp[i + 1][j + 1], dp[i - 1][j + 1]));

        }

      }

    }

    int max = dp[0][0];

    for (int i = 1; i < dp.length; i++) {

      max = Math.max(max, dp[i][0]);

    }

    System.out.println(max);

  }}
```

Targert Subset[True/false]

```
int tar = Integer.parseInt(br.readLine());
boolean[][] dp = new boolean[arr.length + 1][tar + 1];
for (int i = 0; i < dp.length; i++) {
    for (int j = 0; j < dp[0].length; j++) {
        if (i == 0 && j == 0) {
            dp[i][j] = true;
        } else if (i == 0) {
            dp[i][j] = false;
        } else if (j == 0) {
            dp[i][j] = true;
        } else {
            if(dp[i - 1][j] == true){
                dp[i][j] = true;
            } else {
                int val = arr[i - 1];
                if (j >= val && dp[i - 1][j - val] == true) {
                    dp[i][j] = true;
                }
            }
        }
    }
}
System.out.println(dp[dp.length - 1][tar]); }}
```

Coni Change

```
public class Main {
    public static void main(String[] args) throws Exception {
int amt = Integer.parseInt(br.readLine());
        int[] dp = new int[amt + 1];
        dp[0] = 1;
        for(int coin: coins){
            for(int i = 1; i < dp.length; i++){
                if(i >= coin){
                    dp[i] += dp[i - coin];
                }
            }
        }

        System.out.println(dp[amt]);
    }
}
```

Coin Change Permutations

```
public class Main {

    public static void main(String[] args) throws Exception {

int amt = Integer.parseInt(br.readLine());

        int[] dp = new int[amt + 1];

        dp[0] = 1;

        for (int i = 1; i < dp.length; i++) {

            for (int coin : coins) {

                if (i >= coin) {

                    dp[i] += dp[i - coin];

                }

            }

        }


        System.out.println(dp[amt]);

    }

}
```

Zero One Knapsack

```
public class Main {

    public static void main(String[] args) throws Exception {

        int cap = Integer.parseInt(br.readLine());

        int[][] dp = new int[n + 1][cap + 1];

        for (int i = 1; i < dp.length; i++) {

            for(int j = 1; j < dp[0].length; j++){

                int val = values[i - 1];

                int wt = wts[i - 1];



                if(j >= wt){

                    dp[i][j] = Math.max(dp[i - 1][j], dp[i - 1][j - wt] + val);

                } else {

                    dp[i][j] = dp[i - 1][j];

                }

            }

        }



        System.out.println(dp[n][cap]);

    }

}
```

Unbounded Knapsack

```
public class Main {

    public static void main(String[] args) throws Exception {

        int cap = Integer.parseInt(br.readLine());

        int[] dp = new int[cap + 1];

        for (int i = 1; i < dp.length; i++) {

           for(int j = 0; j < wts.length; j++){

               int val = values[j];

               int wt = wts[j];



               if(i >= wt){

                   int factor = dp[i - wt] + val;

                   if(factor > dp[i]){

                       dp[i] = factor;

                   }

               }

           }

        }



        System.out.println(dp[cap]);

    }

}
```

```
public class Main {

    public static void main(String[] args) throws Exception {

        Arrays.sort(items);

        double vib = 0;

        int rc = cap;

  int i = items.length - 1;

        while(i >= 0){

            if(items[i].wt <= rc){

                vib += items[i].val;

                rc -= items[i].wt;

            } else {

                vib += items[i].val * rc / items[i].wt;

                rc = 0;

                break;

            }

            i--;

        }

System.out.println(vib);

    }

public static class Item implements Comparable<Item> {

        int wt;

        int val;

        double vwratio;
```

```
        public int compareTo(Item o){
            if(this.vwratio == o.vwratio){
                return 0;
            } else if(this.vwratio > o.vwratio){
                return +1;
            } else {
                return -1;
            }
        } }   }
```

Count Binary Strings

```
public class Main {
public static void main(String[] args) throws Exception {
      int zeroes = 1;
      int ones = 1;
      for(int i = 2; i <= n; i++){
         int nzeroes = ones;
         int nones = ones + zeroes;
         zeroes = nzeroes;
         ones = nones;
      }
      System.out.println(zeroes + ones);
   }   }
```

Arrange Buildings

```
public class Main {

   public static void main(String[] args) throws Exception {

      BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

      int n = Integer.parseInt(br.readLine());

      long zeroes = 1;

      long ones = 1;

      for (int i = 2; i <= n; i++) {

         long nzeroes = ones;

         long nones = ones + zeroes;

         zeroes = nzeroes;

         ones = nones;

      }

      long oneside = zeroes + ones;

      System.out.println(oneside * oneside);

   }


}
```

Count Encodings

```
public class Main {

    public static void main(String[] args) throws Exception {

        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

        String str = br.readLine();

        int[] dp = new int[str.length()];

        dp[0] = 1;

        for(int i = 1; i < dp.length; i++){

            if(str.charAt(i - 1) == '0'){

                if(str.charAt(i) != '0'){

                    dp[i] = dp[i - 1];

                }

            } else {

                if(str.charAt(i) != '0'){

                    dp[i] = dp[i - 1];

                }

     if(Integer.parseInt(str.substring(i - 1, i + 1)) <= 26){

                    dp[i] += i == 1? 1: dp[i - 2];

                }

            }

        }

     System.out.println(dp[str.length() - 1]);

    }
```

```
public static void printEncodings(String ques, String ans) {
    if (ques.length() == 0) {
        System.out.println(ans);
        return;
    } else if (ques.length() == 1) {
        if (ques.charAt(0) == '0') {
            return;
        } else {
            String ch0 = ques.substring(0, 1);
            String roq0 = ques.substring(1);
            String code0 = (char)('a' + (Integer.parseInt(ch0) - 1)) + "";
            printEncodings(roq0, ans + code0);
        }
    } else {
        if (ques.charAt(0) == '0') {
            return;
        } else {
            String ch0 = ques.substring(0, 1);
            String roq0 = ques.substring(1);
            String code0 = (char)('a' + (Integer.parseInt(ch0) - 1)) + "";
            printEncodings(roq0, ans + code0);
           String ch01 = ques.substring(0, 2);
            String roq01 = ques.substring(2);
            String code01 = (char)('a' + (Integer.parseInt(ch01) - 1)) + "";
```

```
            if (Integer.parseInt(ch01) <= 26) {
                printEncodings(roq01, ans + code01);
            }   }    }
    }
}
Count A+b+c SubSequence
public class Main {
    public static void main(String[] args) throws Exception {
  int acount = 0;
        int bcount = 0;
        int ccount = 0;
        for(int i = 0; i < str.length(); i++){
            if(str.charAt(i) == 'a'){
                acount = 2 * acount + 1;
            } else if(str.charAt(i) == 'b'){
                bcount = 2 * bcount + acount;
            } else if(str.charAt(i) == 'c'){
                ccount = 2 * ccount + bcount;
            }
        }
        System.out.println(ccount);
    }
}
```

```
        long inc = arr[0] < 0 ? 0 : arr[0];
        long exc = 0;
        for (int i = 1; i < arr.length; i++) {
            long ninc = exc + arr[i];
            long nexc = Math.max(inc, exc);
            inc = ninc;
            exc = nexc;
        }
        System.out.println(Math.max(inc, exc));
    }
}
```

Paint House

```
public class Main {

    public static void main(String[] args) throws Exception {

        int[][] arr = new int[n][3];

        for (int i = 0; i < n; i++) {

            String str = br.readLine();

            String[] items = str.split(" ");

            arr[i][0] = Integer.parseInt(items[0]);

            arr[i][1] = Integer.parseInt(items[1]);

            arr[i][2] = Integer.parseInt(items[2]);

        }

        long red = arr[0][0];

        long blue = arr[0][1];

        long green = arr[0][2];

        for (int i = 1; i < arr.length; i++) {

            long nred = arr[i][0] + Math.min(blue, green);

            long nblue = arr[i][1] + Math.min(red, green);

            long ngreen = arr[i][2] + Math.min(red, blue);

            red = nred;

            blue = nblue;

            green = ngreen;

        }

        System.out.println(Math.min(red, Math.min(blue, green)));

    }   }
```

```
public class Main {

    public static void main(String[] args) throws Exception {

        int[][] arr = new int[n][k];

        for (int i = 0; i < n; i++) {

            String str = br.readLine();

            String[] items = str.split(" ");

            for(int j = 0; j < k; j++){

                arr[i][j] = Integer.parseInt(items[j]);

            }

        }

        int min = Integer.MAX_VALUE;

        int smin = Integer.MAX_VALUE;

        for(int j = 0; j < arr[0].length; j++){

            if(arr[0][j] <= min){

                smin = min;

                min = arr[0][j];

            } else if(arr[0][j] <= smin){

                smin = arr[0][j];

            }

        }
```

```
        for (int i = 1; i < arr.length; i++) {
            int cmin = Integer.MAX_VALUE;
            int csmin = Integer.MAX_VALUE;
            for(int j = 0; j < arr[i].length; j++){
                if(arr[i - 1][j] != min){
                    arr[i][j] += min;
                } else {
                    arr[i][j] += smin;
                }
        if(arr[i][j] <= cmin){
                    csmin = cmin;
                    cmin = arr[i][j];
                } else if(arr[i][j] <= csmin){
                    csmin = arr[i][j];
                }
            }
          min = cmin;
            smin = csmin;
        }
        System.out.println(min);
    }
}
```

Paint Fence

```
public class Main {

    public static void main(String[] args) throws Exception {

        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

        int n = Integer.parseInt(br.readLine());

        int k = Integer.parseInt(br.readLine());

        long[] dp = new long[n + 1];

        long same = 0;

        long diff = k;

        dp[1] = same + diff;

        for(int i = 2; i < dp.length; i++){

    same = diff; // number of ways in which this fence is painted same as old fence following the rule of not more than 2.

            diff = dp[i - 1] * (k - 1); // number of ways in which this fence can be painted different from old fence.

            dp[i] = same + diff; // number of ways in which this fence can be painted

        }



        System.out.println(dp[n]);

    }

}
```

Tiling With 2 * 1 Tiles

```
public class Main {

    public static void main(String[] args) throws Exception {

        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

        int n = Integer.parseInt(br.readLine());

        int[] dp = new int[n + 1];

        dp[1] = 1;

        dp[2] = 2;

        for (int i = 3; i <= n; i++) {

            dp[i] = dp[i - 1] + dp[i - 2];

        }

        System.out.println(dp[n]);

    }

}
```

Tiling With M * 1 Tiles

```
public class Main {

    public static void main(String[] args) throws Exception {

        int[] dp = new int[n + 1];

        dp[1] = 1;

        for (int i = 2; i <= n; i++) {

            if (i < m) {

                dp[i] = 1;

            } else if (i == m) {

                dp[i] = 2;

            } else {

                dp[i] = dp[i - 1] + dp[i - m];

            }

        }

        System.out.println(dp[n]);

    }

}
```

Friends Pairing

```
public class Main {

   public static void main(String[] args) throws Exception {

      if (n <= 2) {

         System.out.println(n);

         return;

      }
long[] arr = new long[n + 1];

      arr[0] = 0;

      arr[1] = 1;

      arr[2] = 2;

   for (int i = 3; i <= n; i++) {

         arr[i] = arr[i - 1] + (i - 1) * arr[i - 2];

      }

    System.out.println(arr[n]);

   }


}
```

Partition Into Subsets

```
public class Main {
   public static void main(String[] args) throws Exception {
      BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
      int n = Integer.parseInt(br.readLine());
      int k = Integer.parseInt(br.readLine());
      if (n == 0 || k == 0 || n < k) {
         System.out.println(0);
         return;
      }
      long[][] dp = new long[k + 1][n + 1];
      for (int i = 1; i <= k; i++) {
         for (int j = i; j <= n; j++) {
            if (i == 1 || j == 1 || i == j) {
               dp[i][j] = 1;
            } else {
               dp[i][j] = dp[i - 1][j - 1] + i * dp[i][j - 1];
            }
         }
      }
      System.out.println(dp[k][n]);
   }
}
```

Buy And Sell Stocks - One Transaction Allowed

```
public class Main {

   public static void main(String[] args) throws Exception {

      BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

      int n = Integer.parseInt(br.readLine());

      int[] arr = new int[n];

      for (int i = 0; i < arr.length; i++) {

         arr[i] = Integer.parseInt(br.readLine());

      }

      int msf = arr[0];

      int op = 0;

      for(int i = 1; i < arr.length; i++){

         if(arr[i] < msf){

            msf = arr[i];

         }

         int cp = arr[i] - msf;

         if(cp > op){

            op = cp;

         }

      }

      System.out.println(op);

   }


}
```

Buy And Sell Stocks - Infinite Transactions Allowed

```
public class Main {
   public static void main(String[] args) throws Exception {
int bon = 0;
      int son = 0;
      int op = 0;
      for(int i = 1; i < arr.length; i++){
         if(arr[i] < arr[i - 1]){
            op += arr[son] - arr[bon];
            bon = son = i;
         } else {
            son++;
         }
      }
      op += arr[son] - arr[bon];
      System.out.println(op);
   }
}
```

Buy And Sell Stocks With Transaction Fee - Infinite Transactions Allowed

```
public static void main(String[] args) throws Exception {


      int[] arr = new int[n];
      for (int i = 0; i < arr.length; i++) {
```

```
        arr[i] = Integer.parseInt(br.readLine());
      }
      int fee = Integer.parseInt(br.readLine());
      int bstp = -arr[0];
      int sstp = 0;
      for(int i = 1; i < arr.length; i++){
        int nsstp = 0;
        int nbstp = 0;
        if(sstp - arr[i] > bstp){
          nbstp = sstp - arr[i];
        } else {
          nbstp = bstp;
        }
        if(bstp + arr[i] - fee > sstp){
          nsstp = bstp + arr[i] - fee;
        } else {
          nsstp = sstp;
        }
     bstp = nbstp;
        sstp = nsstp;
      }
      System.out.println(sstp);
    }   }
```

```
  public static void main(String[] args) throws Exception {
    int[] arr = new int[n];
    for (int i = 0; i < arr.length; i++) {
      arr[i] = Integer.parseInt(br.readLine());
    }
    int bstp = -arr[0];
    int sstp = 0;
    int cstp = 0;
    for(int i = 1; i < arr.length; i++){
      int nbstp = 0;
      int nsstp = 0;
      int ncstp = 0;
      if(cstp - arr[i] > bstp){
        nbstp = cstp - arr[i];
      } else {
        nbstp = bstp;
      }
      if(bstp + arr[i] > sstp){
        nsstp = bstp + arr[i];
      } else {
        nsstp = sstp;
      }
```

```
            if(sstp > cstp){
               ncstp = sstp;
            } else {
               ncstp = cstp;
            }

            bstp = nbstp;
            sstp = nsstp;
            cstp = ncstp;
         }

         System.out.println(Math.max(sstp, cstp));
      }

   }
```

Buy And Sell Stocks - Two Transactions Allowed

```
   public static void main(String[] args) throws Exception {
int misf = arr[0];
      int[] ps = new int[arr.length];
      for(int i = 1; i < arr.length; i++){
         if(arr[i] < misf){
            misf = arr[i];
         }
         if(arr[i] - misf > ps[i - 1]){
            ps[i] = arr[i] - misf;
         } else {
            ps[i] = ps[i - 1];
         }
      }
      int masf = arr[arr.length - 1];
      int[] pb = new int[arr.length];
      for(int i = arr.length - 2; i >= 0; i--){
         if(arr[i] > masf){
            masf = arr[i];
         }
         if(masf - arr[i] > pb[i + 1]){
            pb[i] = masf - arr[i];
         } else {
            pb[i] = pb[i + 1];
```

```
            }
        }


        int mp = Integer.MIN_VALUE;
        for(int i = 0; i < arr.length; i++){
            if(ps[i] + pb[i] > mp){
                mp = ps[i] + pb[i];
            }
        }


        System.out.println(mp);
    }


}
```

```
  public static void main(String[] args) throws Exception {
int[][] dp = new int[k + 1][n];
    for(int i = 1; i <= k; i++){
      int fadd = Integer.MIN_VALUE;
      for(int j = 1; j < n; j++){
        if(dp[i - 1][j - 1] - arr[j - 1] > fadd){
          fadd = dp[i - 1][j - 1] - arr[j - 1];
        }

        if(fadd + arr[j] > dp[i][j - 1]){
          dp[i][j] = fadd + arr[j];
        } else {
          dp[i][j] = dp[i][j - 1];
        }
      }
    }

    System.out.println(dp[k][n - 1]);
  }
```