# 1 Handling input and output

How to read white spaces, transform string into int, etc.

# 2 Useful commands

qsort, assert, rounding up and down, etc.

# 3 Basic formulas

Surface, volume,...

# 4 Basic data structures

Graph, stack, queue, priority queue (heap).

# 5 Basic algorithms

## 5.1 Enumerating subsets

```
void enumerate(char a[], int n) {
  int i;
  if (n==0)
    cout << a << "\n"; // print subset
  else {
    enumerate(a,n-1);
    a[n]='-';
    enumerate(a,n-1);
  }
}

int main(void) {
  char a[] = "abcde";
  enumerate(a,5);
  return 0;
}
```

## 5.2 Enumerating permutations

```
void swap(char a[], int i, int j) {
  char t;
  t=a[i]; a[i]=a[j]; a[j]=t;
}

void enumerate(char a[], int n) {
  int i;
  if (n==0)
    cout << a << "\n"; // print permutation
  else
    for (i=0; i<n; i++) {
      swap(a,i,n-1);
      enumerate(a,n-1);
```

```
      swap(a,n-1,i);
    }
}

int main(void) {
  char a[] = "abcde";
  enumerate(a,5);
  return 0;
}
```

# 6 Specifications

## 6.1 World Finals 2002

**Problem A**

**Input format for individual problem:**

| | | |
|---|---|---|
| $n$ | : | $n \in \{1,\ldots,6\}$, number of points |
| $cx1\ cy1\ cz1$ | : | $(cx1, cy1, cz1) \in \mathbb{Z}^3$, coordinates of corner 1, adapt to $\min\{c?1, c?2\}$ |
| $cx2\ cy2\ cz2$ | : | $(cx2, cy2, cz2) \in \mathbb{Z}^3$, coordinates of corner 2, adapt to $\max\{c?1.c?2\}$ |
| $bx[0]\ by[0]\ bz[0]$ | : | $(bx[0], by[0], bz[0]) \in \mathbb{Z}^3$, center of balloon 0 |
| $bx[1]\ by[1]\ bz[1]$ | : | center of balloon 1 |
| ... | : | |
| $bx[n-1]\ by[n-1]\ bz[n-1]$ | : | center of balloon $n-1$ |

Sequence of problems possible, terminated by $n = 0$ (no output).

**Output format:**

| | | |
|---|---|---|
| Box 1: $v$ | : | $v \in \mathbb{R}_+$: volume in box not enclosed by the balloons |
| Box 2: $v$ | : | |
| ... | : | |

**Problem:** Find $S \subset \{0,\ldots,n-1\}$ s.t. $\sum_{i\in S} \frac{4}{3}\pi r[i]^3$ is maximized.

**Conditions:**

- For all $i, j \in S$: $\sqrt{(bx[i]-bx[j])^2 + (by[i]-by[j])^2 + (bz[i]-bz[j])^2} \geq r[i] + r[j]$ (no intersection of balloons).
- For all $i$: $cx1 \leq bx[i] - r[i]$ and $bx[i] + r[i] \leq cx2$ and $cy1 \leq by[i] - r[i]$ and $by[i] + r[i] \leq cy2$ and $cz1 \leq bz[i] - r[i]$ and $bz[i] + r[i] \leq cz2$ (no balloon intersects box).

**Method:** Enumerate all subsets $S$ of $\{0,\ldots,n-1\}$, enumerate all permutations $\pi$ of $S$. For each $\pi$, compute for $i = 0$ to $|S|-1$ the maximum $r[i]$ that fulfills the conditions for $j < i$.

**Problem B**

**Input format for individual problem:**

| | | |
|---|---|---|
| $m$ | : | $m \in \{1,\ldots,20\}$: number of codewords |
| $w[0]$ | : | $w[0] \in \{0,1\}^k$ for some $k \in \{1,\ldots,20\}$: codeword 0 |
| ... | : | ... |
| $w[m-1]$ | : | codeword $m-1$ |

Sequence of problems possible, terminated by $m = 0$ (no output).

**Output format:**

| | | |
|---|---|---|
| Code 1: $b$ bits | : | $b \in Z_+$: length of shortest encoding that is not uniquely encodable |
| $word[0]$ | : | $word[0]$: shortest encoding itself, first 20 bits (or remaining bits) |
| $word[1]$ | : | $word[1]$: shortest encoding itself, second 20 bits (or remaining bits) |
| | : | |
| Code 2: ... | : | |

**Problem:** Find length of shortest encoding and shortest encoding itself.

**Conditions:**

- Code is always guaranteed not to be uniquely decodable.

**Method:** Starting with $k = 1$, compute the set of all codes of length $k$ with the recursive formula

$$Code(k) = \bigcup_{i=1}^{k-1} Code(i) \circ Code(k-i)$$

For each constructed codeword, check if it is already in the set $Code(k)$. If so, the shortest ambiguous codeword has been found. Output its length and itself.

**Problem C**

**Input format for individual problem:**

| | | |
|---|---|---|
| $n$ $c$ | : | $n \in \{2, \ldots, 20\}$: number of locations |
| | : | $c \in \mathbb{R}_+$: capacity in units of food and water |
| $x[0]$ $y[0]$ | : | $(x[0], y[0]) \in \mathbb{Z}^2$: coordinates of location 0, starting point |
| $x[1]$ $y[1]$ | : | $(x[1], y[1])$: coordinates of location 1 |
| ... | : | ... |
| $x[n-1]$ $y[n-1]$ | : | $(x[n-1], y[n-1])$: coordinates of location $n-1$, destination |

Sequence of problems possible, terminated by $n = 0$ and $c = 0$ (no output).

**Output format:**

| | | |
|---|---|---|
| Trial 1: $f$ units of food | : | $f \in \{1, \ldots, 1000000\}$: minimum number of units of food needed |
| | : | |
| Trial 2: Impossible | : | or impossible |

**Problem:** Compute the minimum number of units $f$ needed for the trip from the starting point to the destination.

**Conditions:**

- Food can only be acquired at starting point.
- Each oasis has infinite water supply and can be used to store food.
- To go from $(x1, y1)$ to $(x2, y2)$, you need $\sqrt{(x1-x2)^2 + (y1-y2)^2}$ units of food and water.
- Number of food units and water units cannot exceed capacity.

**Method:** Use BFS from destination. Consider an edge $(v, w)$ of length $d$. To accumulate a food supply of $f_w > 0$ at $w$ via $v$, we use the following rules:

if $(2d \geq c)$ then *impossible* // *unreachable*
else
    if $(f_w + 2d \leq c)$ then $f_v = f_w + 2d$ // *in one walk*
    else
        if $(3d \geq c)$ then *impossible* // *no return*
        else
            $t = \lfloor (c - 3d)/f_w \rfloor$ // *number of round trips*
            $f_v = (c - d)t + (f_w + d - (c - 3d)t)$

Now, initially we set $f_d = 0$ for the destination $d$ and all other nodes have $f_v = \infty$. Then we perform $n$ iterations. In each iteration, we take every node $w$ with current $f_w < \infty$ and compute for every $v$ the food $f'_v$ necessary for $(v, w)$ (see the rules above). If $f'_v < f_v$, we set $f_v = f'_v$. If after $n$ iterations, $f_s = \infty$ for the source, we output "impossible". Otherwise, we output $\lceil f_s \rceil$.

**Problem E**

**Input format for individual problem:**

| | | |
|---|---|---|
| $n$ | : | $n \in \{1, \ldots, 50\}$: number of islands |
| $x[0]$ $y[0]$ $m[0]$ | : | $(x[0], y[0]) \in \mathbb{Z}^2$: coordinate of island 0, the main island |
| | : | $m[0] \in \mathbb{Z}_+$: number of inhabitants of island 0 |
| $x[1]$ $y[1]$ $m[1]$ | : | $(x[1], y[1])$: coordinate of island 1 |
| $\ldots$ | : | $\ldots$ |
| $x[n-1]$ $y[n-1]$ $m[n-1]$ | : | $(x[n-1], y[n-1])$: coordinate of island $n-1$ |

Sequence of problems possible, terminated by $n = 0$ (no output).

**Output format:**

| | | |
|---|---|---|
| Island Group: 1 Average $d$ | : | $d \in \mathbb{R}_+$, form $?.xx$: average number of days until inhabitant connected |
| | : | |
| Island Group: 2 Average $d$ | : | |

**Problem:** Minimize average connection time, i.e. given that $t_i$ is the time when island $i$ is connected, minimize

$$\frac{\sum_i t_i \cdot m_i}{\sum_i m_i}$$

**Conditions:**

- Total amount of cable used is minimal.
- Construction on all cable links starts at the same time.
- A cable link is constructed at rate 1 km/day.

**Method:** Use recursive MST algorithm that adds a minimum edge connecting two different trees in each recursion. If there is more than one, recurse to each one of the alternatives. Once the MST has been constructed, compute the average connection time, where $t_i$ is the length of the longest edge in the MST from island $i$ to island 0.

**Problem F**

**Input format for individual problem:**

| | | |
|---|---|---|
| $n$ | : | $n \in \{3, \ldots, 100\}$: number of points on polygon |
| $x[0]$ $y[0]$ $h[0]$ | : | $(x[0], y[0]) \in \mathbb{Z}^2$: coordinates of point 0 |
| | : | $h[0] \in \{0, 1\}$: $h[0] = 1$, then there is a hole |
| $x[1]$ $y[1]$ $h[1]$ | : | coordinates of point 1 |
| $\ldots$ | : | |
| $x[n-1]$ $y[n-1]$ $h[n-1]$ | : | coordinates of point $n-1$ |

Sequence of problems possible, terminated by $n = 0$ (no output).

**Output format:**

Cave 1: Oil capacity = $oc$ : $oc \in \mathbb{Z}_+$, oil capacity rounded to nearest integer
:
Cave 2: Oil capacity = $oc$ :

**Problem:** Compute the maximum oil capacity the cave can hold.

**Conditions:**

- Oil level in every connected section can only be up to hole.

**Method:** For each connected component, the following representation is maintained: $p_1, p_2, p_3, p_4, \ldots, p_{2k-1}, p_{2k}$, where every $(p_{2i-1}, p_{2i})$ represents a segment of the component. Also, a variable $h_c \in \{0, 1\}$ is kept that indicates whether the component has already had a hole and a variable $y_c$ is kept indicating up to which height the component has already been considered.

Sort points according to their $y$ values, with smallest $y$ in front. Start with the lowest point. Take these points one by one. For each point $p$ do the following:

If there is no component $c$ so that $p$ is in the area spanned by $\ell(p_{2i-1})$ and $\ell(p_{2i})$, then start a new component with $(p, p)$ and otherwise replace $(p_{2i-1}, p_{2i})$ by $(p'_{2i-1}, p)$ and $(p, p'_{2i})$ where $p'_{2i-1}$ and $p'_{2i}$ are adjusted to $p$'s $y$-coordinate. Adapt also all other points in that component and compute the area covered in the adaptation. Add this area to $oc_c$ if $h_c = 0$. If $h_p = 1$, then set $h_c = 1$.

**Problem G**

**Input format for individual problem:**

Sequence of problems possible, terminated by $n = 0$ and $c = 0$ (no output).

**Output format:**

**Problem:**

**Conditions:**

-

**Method:**