1. (a) Express $f_i(\triangle^d)$, $f_i(\square^d)$, and $f_i(\diamond^d)$ as simple functions of $i$ and $d$.

**Solution:** The $d$-simplex $\triangle^d$ is the convex hull of $d+1$ affinely independent points. It's not hard to see that every subset of $d$ of those points spans a facet of $\triangle^d$, and that facet is a $(d-1)$-simplex. Thus, by induction, *every* subset of the $d+1$ points spans a face of the simplex. In particular, every subset of $i+1$ vertices spans an $i$-dimensional face.

$$\boxed{f_i(\triangle^d) = \binom{d+1}{i+1}}$$

In particular, the total number of faces is $F(\triangle^d) = 2^{d+1}$. In fact, the Hasse diagram of the face lattice of $\triangle^d$ is isomorphic to the edges of the hypercube $\square^{d+1}$!

The $d$-cube $\square^d = [-1,1]^d$ has exactly $2d$ facets, each a $(d-1)$-cube lying on a hyperplane of the form $x_i = \pm 1$. Every face of dimension $i$ is the intersection of $d-i$ facets, each in a different coordinate direction, but with completely independent signs. The only exception to this characterization is that there is one $-1$-face: the empty set.

$$\boxed{f_i(\square^d) = 2^{d-i}\binom{d}{i} + [i = -1]}$$

In particular, the total number of faces is $F(\square^d) = 3^d + 1$.

The $d$-cross-polytope $f_i(\diamond^d)$ has $2^d$ facets, each a $(d-1)$-simplex whose vertices have the form $(\pm 1, 0, \ldots 0)$, $(0, \pm 1, \ldots, 0)$, $\ldots$, $(0, 0, \ldots, \pm 1)$, where the signs are chosen independently. Thus, by the analysis of simplices, each $i$-dimensional face is chosen by taking a subset of $i+1$ coordinate directions, each with its own sign. The only exception to this characterization is that there is one $d$-face: the entire polytope.

$$\boxed{f_i(\diamond^d) = 2^{i+1}\binom{d}{i+1} + [i = d]}$$

In particular, the total number of faces is $F(\diamond^d) = 3^d + 1$. ■

(b) Express $f_i(P \times Q)$ as a function of various $f_j(P)$ and $f_k(Q)$.

**Solution:** Like every polytope, $P \times Q$ has exactly one empty face. For all $i \geq 0$, every $i$-face of $P \times Q$ is the product of a *non-empty* $j$-face of $P$ and a *non-empty* $(i-j)$-face of $Q$, for some integer $j$.

$$\boxed{f_i(P \times Q) = [i = -1] + \sum_{j=0}^{i} f_j(P) \cdot f_{i-j}(Q)}$$

(This formula exploits the fact that $f_i(X) = 0$ for all $i > \dim X$.) In particular, we have the following formulas for the number of vertices, facets, and faces.

$$\begin{aligned} f_0(P \times Q) &= f_0(P) \cdot f_0(Q) \\ f_{d+e-1}(P \times Q) &= f_{d-1}(P) + f_{e-1}(Q) \\ F(P \times Q) &= (F(P) - 1) \cdot (F(Q) - 1) + 1 \end{aligned}$$

■

(c) Express $f_i(P \oplus Q)$ as a function of various $f_j(P)$ and $f_k(Q)$.

**Solution:** Like every polytope, $P \oplus Q$ has exactly one full-dimensional face. For all $i < d + e$, every $i$-face of $P \times Q$ is the convex hull of a *proper* $j$-face of $P$ and a *proper* $(i - j - 1)$-face of $Q$, for some integer $j$. A proper face is any face except the entire polytope; to enforce this condition, we require that $j < d$ and $i - j - 1 < e$ (so $j \geq i - e$).

$$\boxed{f_i(P \oplus Q) = [i = d + e] + \sum_{j=i-e}^{d-1} f_j(P) \cdot f_{i-j-1}(Q)}$$

(This formula exploits the fact that $f_i(X) = 0$ for all $i < -1$.) In particular, we have the following formulas for the number of vertices, facets, and faces.

$$\begin{aligned} f_0(P \oplus Q) &= f_0(P) + f_0(Q) \\ f_{d+e-1}(P \oplus Q) &= f_{d-1}(P) \cdot f_{e-1}(Q) \\ F(P \oplus Q) &= (F(P) - 1) \cdot (F(Q) - 1) + 1 \end{aligned}$$

■

(d) Why are the answers to parts (b) and (c) so similar?

**Solution:** Cartesian products and orthogonal sums are duals of each other. This follows from the general duality between convex hulls of points and intersections of halfspaces. Specifically, we can define the product $P \times Q$ as the intersection of two polyhedra:

$$P \times Q = (P \times \mathbb{R}^e) \cap (\mathbb{R}^d \times Q)$$

Any supporting hyperplane of $P \times \mathbb{R}^e$ dualizes to a point inside $P^* \times 0^e$ (where $0^e$ is the vector of $e$ zeros, or the origin of $\mathbb{R}^e$). Thus,

$$\begin{aligned} (P \times Q)^* &= ((P \times \mathbb{R}^e) \cap (\mathbb{R}^d \times Q))^* \\ &= \text{conv}\{(P \times \mathbb{R}^e)^* \cup (\mathbb{R}^d \times Q)\}^* \\ &= \text{conv}\{(P^* \times 0^e) \cup (0^d \times Q^*)\} \\ &= P^* \oplus Q^* \end{aligned}$$

The last line is just the definition of orthogonal sum. Dualizing a polytope inverts its face lattice, mapping facets to vertices, ridges to edges, the whole polytope to the empty set, and so on:

$$f_i(X^*) = f_{\dim X - i - 1}(X).$$

Thus, the $f$-vectors of Cartesian products and orthogonal sums are related as follows:

$$\boxed{f_i(P \times Q) = f_i((P^* \oplus Q^*)^*) = f_{d+e-i-1}(P^* \oplus Q^*)}$$

We can easily verify that the formulas from parts (a) and (b) satisfy this equation. ■

(e) Express $f_i(P * Q)$ as a function of various $f_j(P)$ and $f_k(Q)$.

**Solution:** Every $i$-face of the $P * Q$ is the join of a $j$-face of $P$ and an $(i-j)$-face of $Q$; these faces need not be proper or empty.

$$\boxed{f_i(P * Q) = \sum_{j+k=i} f_j(P) \cdot f_k(Q)}$$

In particular, the following formulas count the number of vertices, facets, and faces.

$$f_0(P * Q) = f_0(P) + f_0(Q)$$
$$f_{d+e-1}(P * Q) = f_{d-1}(P) + f_{e-1}(Q)$$
$$F(P * Q) = F(P) \cdot F(Q)$$

Unlike Cartesian products and orthogonal sums, which are duals of each other, the join operator is actually its own dual: $P^* * Q^* = (P * Q)^*$. This is not entirely obvious, but it essentially follows from the fact that any two affinely independent subspaces of $\mathbb{R}^{d+e+1}$ are dual to each other (under some duality map). ■

2. Let $P$ be a convex polygon in the plane with $n$ vertices $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$ in counterclockwise order, and let $z_1, z_2, \ldots, z_n$ be arbitrary real numbers. Describe and analyze a randomized algorithm to construct the convex hull of the sequence of points $(x_1, y_1, z_1), (x_2, y_2, z_2), \ldots, (x_n, y_n, z_n)$ in $O(n)$ expected time.

**Solution:** We can use a variant of the randomized incremental algorithm presented in class. We insert the points one at a time in random order. For each new point $p_i$, we somehow compute a single *edge* on the previous convex hull $H_{i-1}$ that is visible to $p_i$, compute and remove all the facets of $H_{i-1}$ that are visible to $p_i$ (using depth-first search, starting from the initial edge), and then join the resulting hole to $p_i$ to obtain the new convex hull $H_i$.

By Euler's formula, $H_i$ has at most $2i - 4$ facets, and any of its $i$ vertices is equally likely to be the last vertex inserted. Thus, on average, $H_i$ has just under 6 new facets that did not belong to $H_{i-1}$. Thus, during the entire algorithm, we expect to create less than $6n$ facets. And since we destroy fewer facets than we create, we expect to destroy less than $6n$ facets.

Let $e_i$ be the magically-determined edge of $H_{i-1}$ visible to $p_i$. One of the two facets adjacent to $e_i$ must be visible to $p_i$, and for each visible facet, the depth-first search examines at most three invisible facets. Thus, the time taken by the depth-first search is at most a constant times the number of visible facets found. Therefore, ignoring the cost of finding the initial visible edges, the running time of the algorithm is $O(n)$.

So to get an $O(n)$ time bound, we need to find each visible edge in $O(1)$ time. To make this possible, we add a preprocessing phase that precomputes these visible vertices in $O(n)$ time. The preprocessing phase is essentially a *backwards* randomized incremental algorithm for constructing the two-dimensional convex hull of the $x$- and $y$-coordinates. We can actually run this backwards algorithm precisely because we are given the "shadow hull" as part of the input!

Here is the algorithm in more detail. The first two loops are the preprocessing phase; notice that we need no new data structures more complicated than an array. The very first line computes a random permutation, which we use twice: once in reverse order during preprocessing, and once forward while we build the hull.

```
ConvexConvexHull(P[0..n-1]):
  π[0..n-1] ← random permutation of [0..n-1]

  ⟨⟨Preprocess visible edges⟩⟩
  for i ← 0 to n-1
      next[i] ← i+1 mod n
      prev[i] ← i-1 mod n
  for i ← n-1 downto 0
      next[prev[π[i]]] ← next[π[i]]
      prev[next[π[i]]] ← prev[π[i]]

  ⟨⟨Now incrementally build the hull⟩⟩
  H ← conv{P[π[0]], P[π[1]], P[π[2]], P[π[3]]}
  for i ← 4 to n-1
      ⟨P[π[i]] can see the edge P[next[π[i]]], P[prev[π[i]]]⟩
      find and delete all facets of H visible to P[π[i]] using DFS
      join P[π[i]] to edges of the resulting hole in H
```

Alternately, we can describe the same algorithm recursively, as follows.

```
RecConvexConvexHull(P):
  Choose p uniformly at random from P
  H ← RecConvexConvexHull(P \ {p})
  ⟨p can see the edge between its neighbors in P⟩
  find and delete all facets of H visible to p using DFS
  join p to edges of the resulting hole in H
  return H
```

This randomized algorithm was discovered by Chew in an otherwise-unpublished technical report [2], simplifying an earlier deterministic algorithm of Aggarwal *et al.* [1]. Chew's paper was probably the first to analyze randomized incremental algorithms as through they were running backwards. The recursive formulation of Chew's algorithm was described by Seidel in an excellent survey on backwards analysis [3].

[1] L. Paul Chew. Building Voronoi diagrams for convex polygons in linear expected time. Technical Report PCS-TR90-147, Department of Mathematics and Computer Science, Dartmouth College, 1986.

[2] Alok Aggarwal, Leonidas J. Guibas, James B. Saxe, and Peter W. Shor. A linear-time algorithm for computing the Voronoi diagram of a convex polygon. *Discrete & Computational Geometry* 4(6):591–604, 1989.

[3] Raimund Seidel. Backwards Analysis of Randomized Geometric Algorithms. *New Trends in Discrete and Computational Geometry* (János Pach, editor), pp. 37–68. Algorithms and Combinatorics 10, Springer-Verlag, 1993.

■

3. (a) Given a set of $n$ points in general position in the plane, *exactly* how many vertices does its Voronoi diagram have?

**Solution:** Let $v$ and $e$ denote the number of Voronoi vertices and edges, respectively, of a set $P$ of $n$ points in the plane. Assuming general position, each Voronoi vertex lies on exactly three edges. Conversely, most Voronoi edges contains exactly two vertices, but there are a few Voronoi edges with only one vertex, one for each edge of the convex hull of $P$. Thus, the number of incident vertex-edge pairs is exactly $\boxed{3v = 2e - h}$, where $h$ is the number of extreme points in $P$.

We also need the following version of Euler's formula: $\boxed{v - e + n = 1}$, where $n$ is the number of faces. We can prove this by induction by simplifying the graph, without changing the quantity $v - e + n$. First, we can delete any edge that separates two faces of the graph, thereby deceasing both $e$ and $n$ by 1. Once the graph has a single face, it consists of one or more trees, each with exactly one infinite edge. At this point, we can delete any vertex with degree 1 and its only edge, decreasing both $e$ and $v$ by 1. In the end, we are left with an empty graph, which has one face (the entire plane).

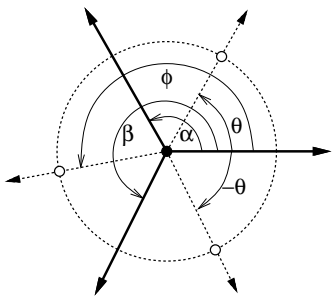
Combining these two formulas, we have

$$\begin{aligned} & 3v = 2e - h = 3 + 3e - 3n \\ \Longrightarrow \quad & e = 3n - h - 3 \\ \Longrightarrow \quad & v = (2e - h)/3 = \boxed{2n - h - 2} \end{aligned}$$

■

(b) Describe an algorithm to reconstruct a set of $n$ points from its Voronoi diagram in $O(n)$ time. What does your algorithm do if the input is not actually a Voronoi diagram? *[Hint: First solve the special cases $n = 3$ and $n = 4$.]*

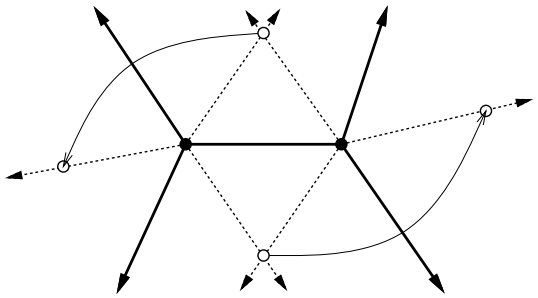
**Solution:** First consider the special case $n = 3$. The Voronoi diagram consists of three rays not all in the same halfplane. In this case, there is a one-parameter family of solutions, where the parameter is the distance form the sites to their common Voronoi vertex. Varying this parameter scales the points outwards from the Voronoi vertex, so we can describe the entire family of solutions by three more rays.




Reconstructing (dotted) site rays from three (solid) Voronoi edges

Without loss of generality, suppose one of the Voronoi edges points in the positive $x$-direction. Let $\alpha$ and $\beta$ denote the angles of the other two Voronoi edges, and let $\theta$, $\phi$, and $-\theta$ denote the angles of the site rays, as illustrated above. The definition of Voronoi diagrams implies immediately that $\alpha = (\theta + \phi)/2$ and $\beta = (\phi + 2\pi - \theta)/2$. Equivalently, we have $\boxed{\theta = \alpha - \beta + \pi}$ and $\boxed{\phi = \alpha + \beta + \pi}$.

Now consider the case $n = 4$. By part (a), the Voronoi diagram has at least two vertices and at least one finite edge. For each Voronoi vertex, we can determine a triple of rays by the argument above. Pick some finite For each finite Voronoi edges, two pairs of these rays intersect at the only possible locations for those two sites. To find the other two sites, we merely reflect each known site across one of its neighboring Voronoi edges.



Reconstructing sites rays from four Voronoi edges

For larger values of $n$, we apply the $n = 4$ technique to any finite Voronoi edge, and then construct the remainder of the sites by a series of reflections. This clearly requires only $O(n)$ time. If the input is *not* actually a Voronoi diagram, then different sequences of reflections will lead to different point sets. Thus, we can check whether the input is really a Voronoi diagram by reflecting across *every* edge and checking for consistency, still taking only $O(n)$ time overall. ■

*(c) Now suppose we want to reconstruct a set of $n$ points given only the $v$ *vertices* of its Voronoi diagram and the integer $n$. Can we reconstruct any set of $n$ points from its $v$ Voronoi vertices if $v \geq n$?

**Solution:** I believe you can, but I don't have a proof. ■