600.120 Intermediate Programming, Spring 2017*

Misha Kazhdan

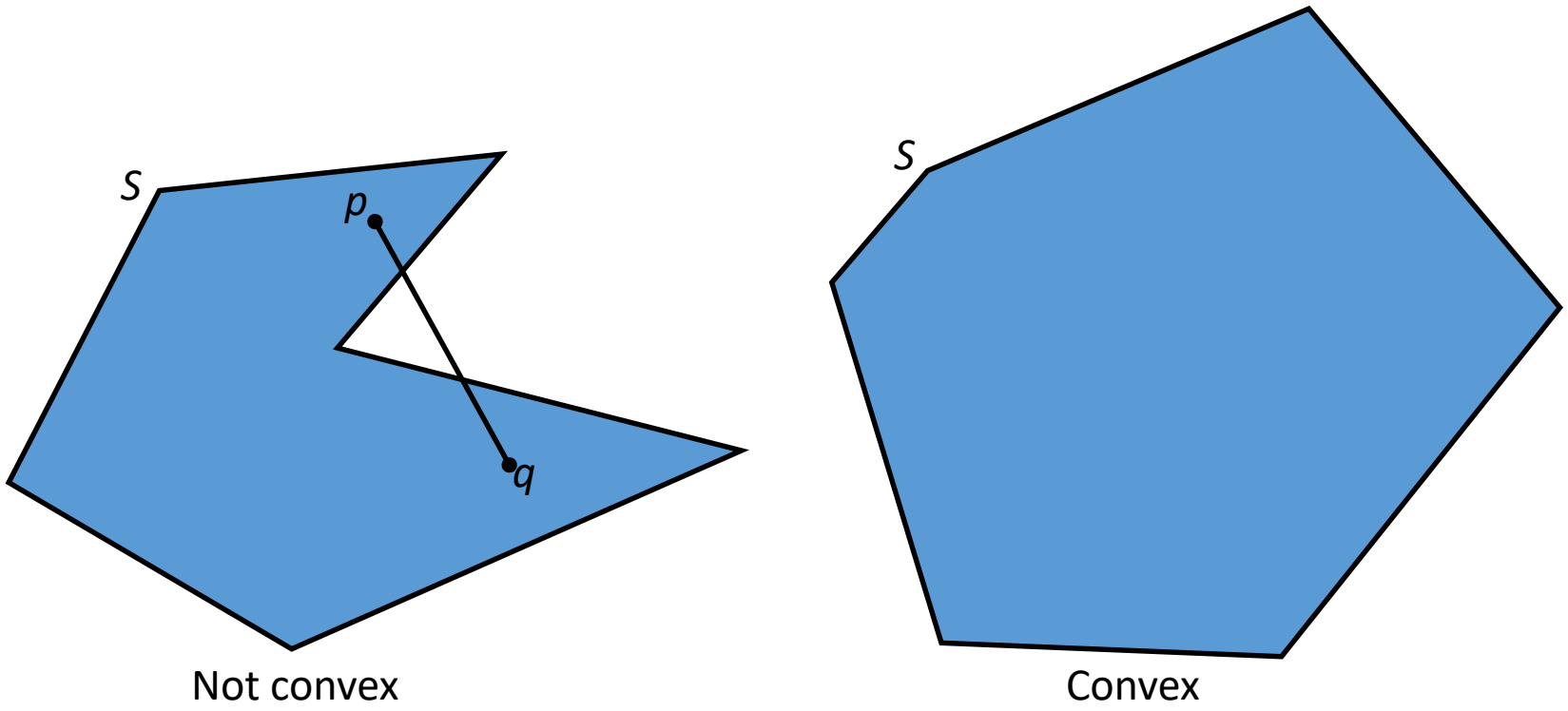
**Much of the code in these examples is not commented because it would otherwise not fit on the slides. This is bad coding practice in general and you should not follow my lead on this.*

Outline

- Convex hulls
- Midterm stuff

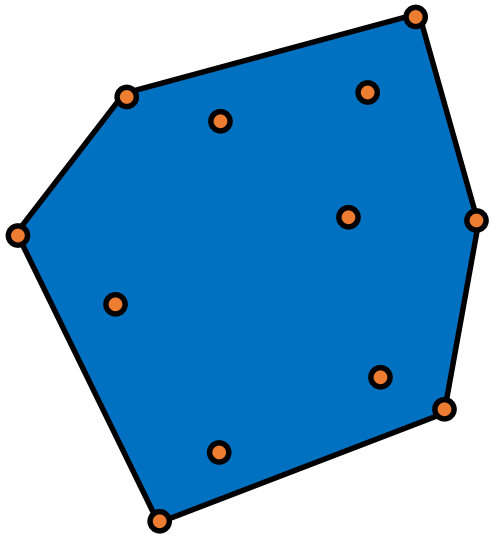
Convexity

A set is *convex* if for any two points in the set the line segment between the points is also inside the set



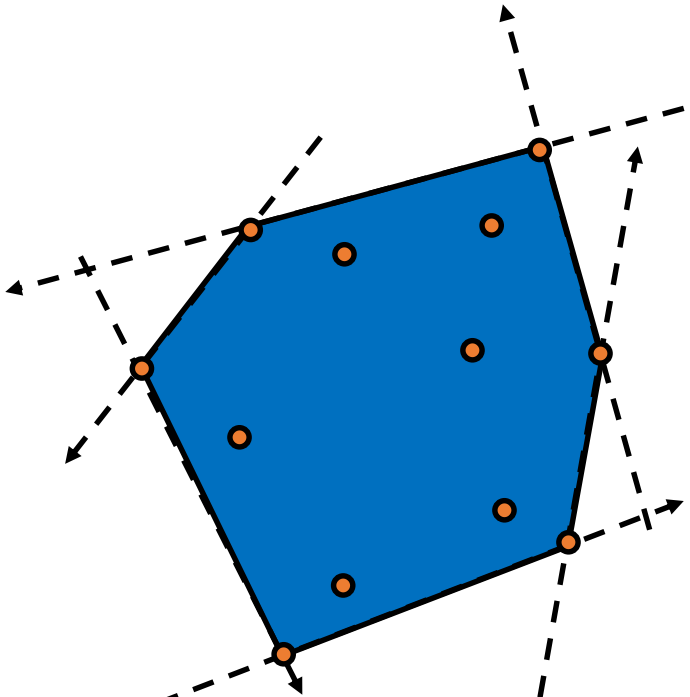
Convex Hulls

- Given a set of points, the *convex hull* is the smallest convex set that contains the points
  - It is a polygon with a subset of the points as its vertices



Convex Hulls

- Given a set of points, the *convex hull* is the smallest convex set that contains the points
  - It is a polygon with a subset of the points as its vertices
  - For each pair of successive vertices, the remaining points are to the left*



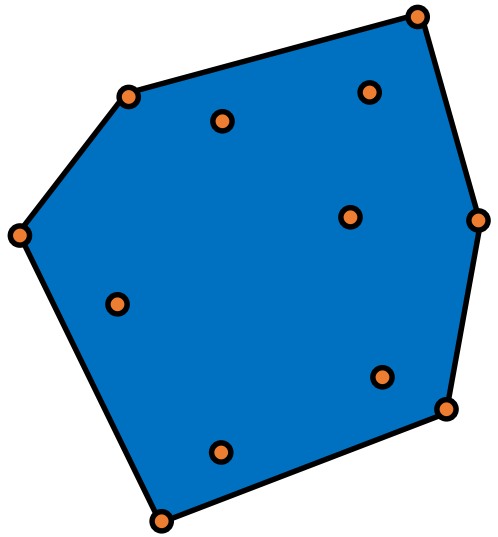
*Assuming the vertices are oriented counter-clockwise

Convex Hulls

Q [Algorithm]: How to compute the convex hull of points in 2D?

Q [Data-Structure]: How to represent the convex hull of points in 2D?

- Often, the choice of algorithm informs the choice of data-structure

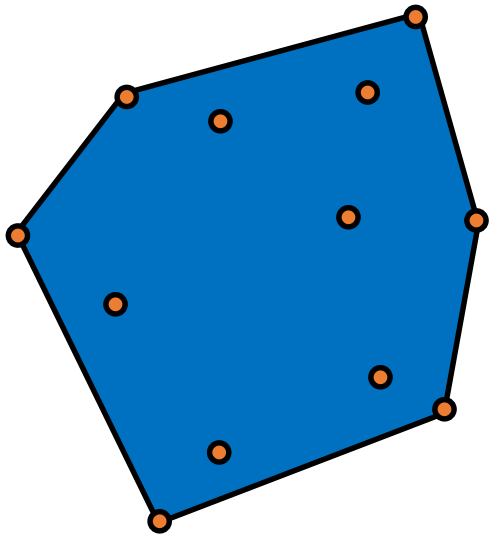


Convex Hulls

Q [Algorithm]: How to compute the convex hull of points in 2D?

A [Algorithm]: Incremental algorithm

1. Solve the problem for a small subset of the input
2. Use the solution for the small subset to get the solution for a larger subset
3. Keep growing until you've solved the problem for the whole dataset

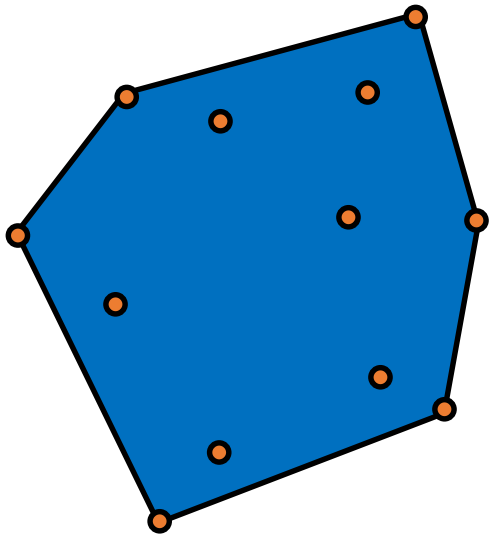


Convex Hulls

Q [Algorithm]: How to compute the convex hull of points in 2D?

A [Algorithm]: Incremental algorithm

1. Construct the (convex) triangle through the first three points
2. Consider the remaining points one at a time
   A. If a point is inside the current hull, ignore it
   B. Otherwise, add it to the hull

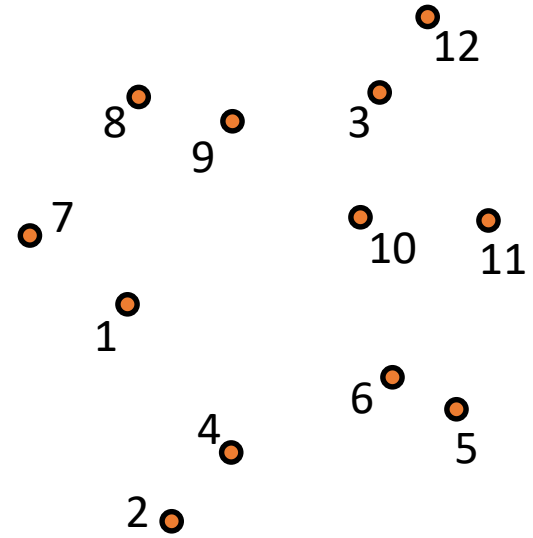


Convex Hulls

Q [Algorithm]: How to compute the convex hull of points in 2D?

A [Algorithm]: Incremental algorithm

1. Construct the (convex) triangle through the first three points
2. Consider the remaining points one at a time
   A. If a point is inside the current hull, ignore it
   B. Otherwise, add it to the hull

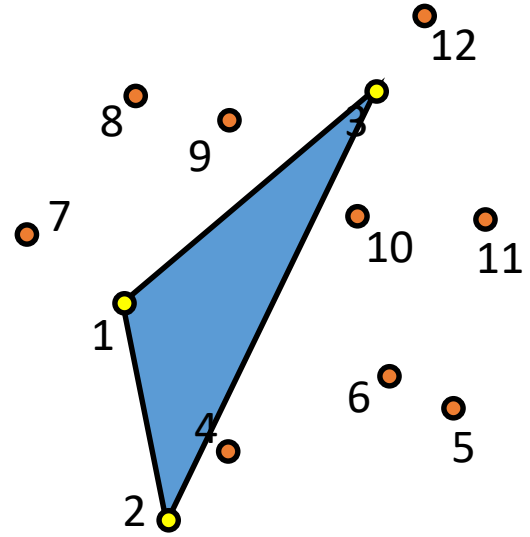


Convex Hulls

Q [Algorithm]: How to compute the convex hull of points in 2D?

A [Algorithm]: Incremental algorithm

1. **Construct the (convex) triangle through the first three points**
2. Consider the remaining points one at a time
   A. If a point is inside the current hull, ignore it
   B. Otherwise, add it to the hull

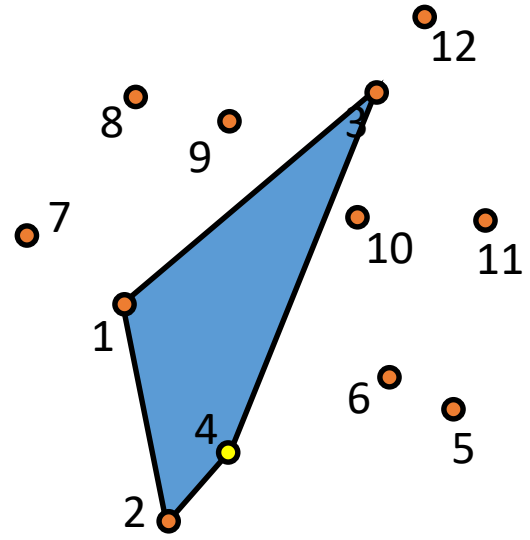


Convex Hulls

Q [Algorithm]: How to compute the convex hull of points in 2D?

A [Algorithm]: Incremental algorithm

1. Construct the (convex) triangle through the first three points
2. Consider the remaining points one at a time
   A. If a point is inside the current hull, ignore it
   B. Otherwise, add it to the hull

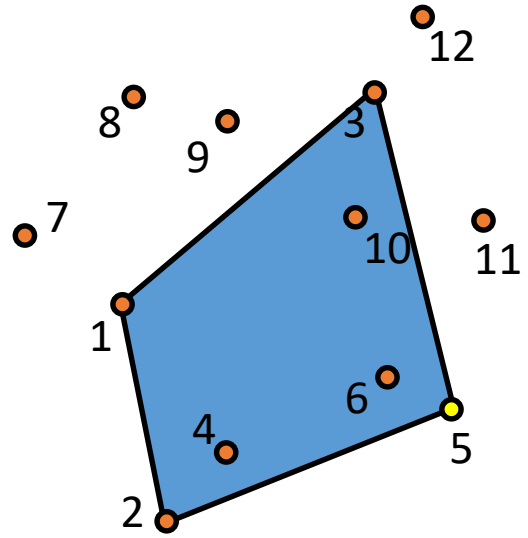


Convex Hulls

Q [Algorithm]: How to compute the convex hull of points in 2D?

A [Algorithm]: Incremental algorithm

1. Construct the (convex) triangle through the first three points
2. Consider the remaining points one at a time
   A. If a point is inside the current hull, ignore it
   B. Otherwise, add it to the hull

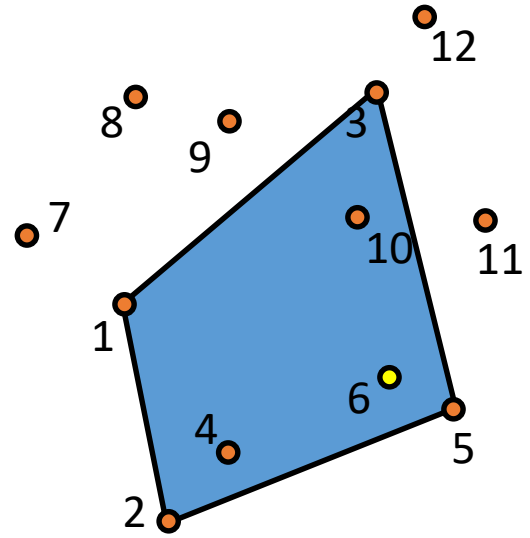


Convex Hulls

Q [Algorithm]: How to compute the convex hull of points in 2D?

A [Algorithm]: Incremental algorithm

1. Construct the (convex) triangle through the first three points
2. Consider the remaining points one at a time
   - **A. If a point is inside the current hull, ignore it**
   - B. Otherwise, add it to the hull

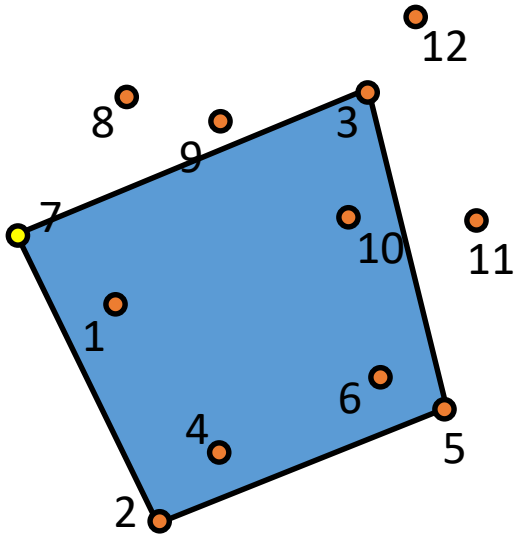


Convex Hulls

Q [Algorithm]: How to compute the convex hull of points in 2D?

A [Algorithm]: Incremental algorithm

1. Construct the (convex) triangle through the first three points
2. Consider the remaining points one at a time
   A. If a point is inside the current hull, ignore it
   B. Otherwise, add it to the hull

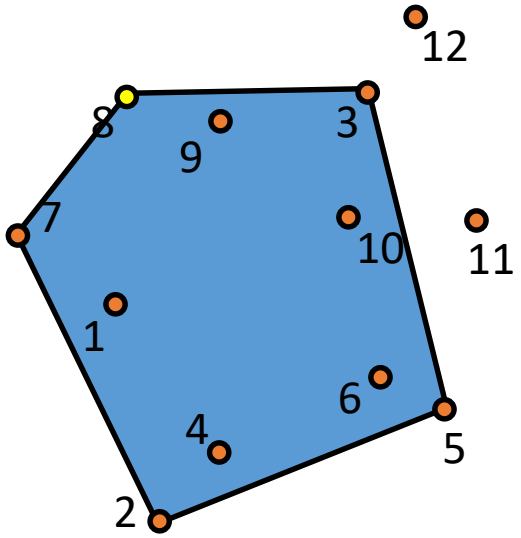


Convex Hulls

Q [Algorithm]: How to compute the convex hull of points in 2D?

A [Algorithm]: Incremental algorithm

1. Construct the (convex) triangle through the first three points
2. Consider the remaining points one at a time
   A. If a point is inside the current hull, ignore it
   B. Otherwise, add it to the hull

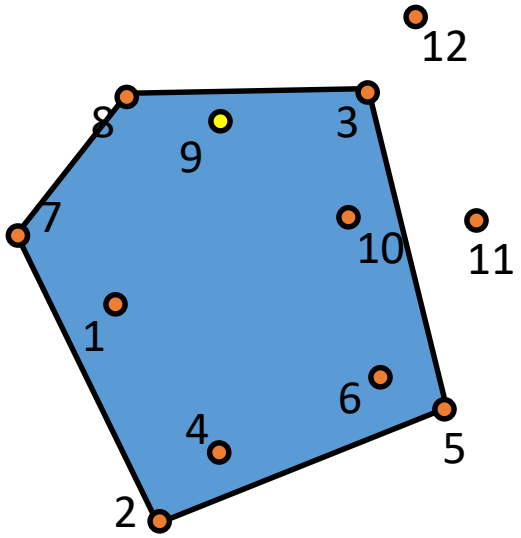


Convex Hulls

Q [Algorithm]: How to compute the convex hull of points in 2D?

A [Algorithm]: Incremental algorithm

1. Construct the (convex) triangle through the first three points
2. Consider the remaining points one at a time
   - **A. If a point is inside the current hull, ignore it**
   - B. Otherwise, add it to the hull

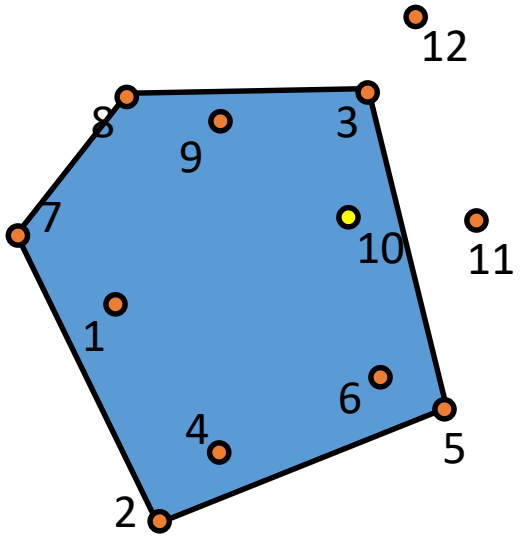


Convex Hulls

Q [Algorithm]: How to compute the convex hull of points in 2D?

A [Algorithm]: Incremental algorithm

1. Construct the (convex) triangle through the first three points
2. Consider the remaining points one at a time
   - **A. If a point is inside the current hull, ignore it**
   - B. Otherwise, add it to the hull

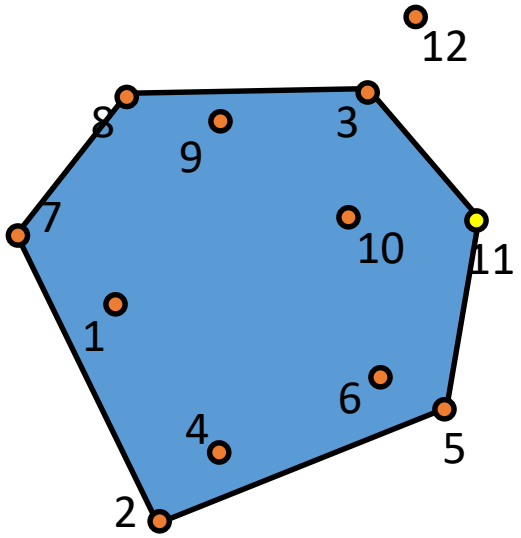


Convex Hulls

Q [Algorithm]: How to compute the convex hull of points in 2D?

A [Algorithm]: Incremental algorithm

1. Construct the (convex) triangle through the first three points
2. Consider the remaining points one at a time
   A. If a point is inside the current hull, ignore it
   B. Otherwise, add it to the hull

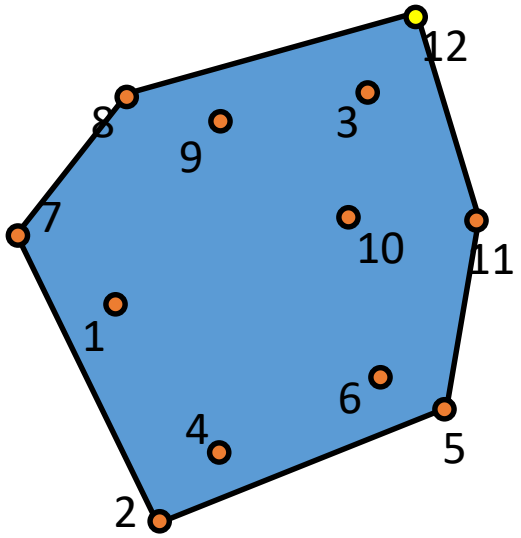


Convex Hulls

Q [Algorithm]: How to compute the convex hull of points in 2D?

A [Algorithm]: Incremental algorithm

1. Construct the (convex) triangle through the first three points
2. Consider the remaining points one at a time
   A. If a point is inside the current hull, ignore it
   B. Otherwise, add it to the hull

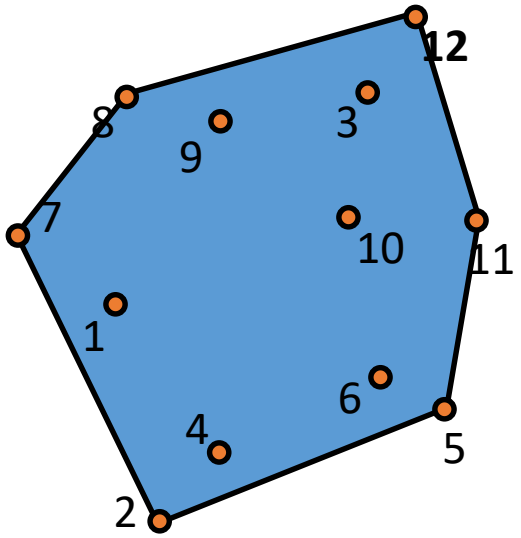


Convex Hulls

Q [Data-Structure]: How to represent the convex hull of points in 2D?

A [Data-Structure]: Circular linked-lists

- The representation of the polygon must support dynamic insertion / deletion



Convex Hulls

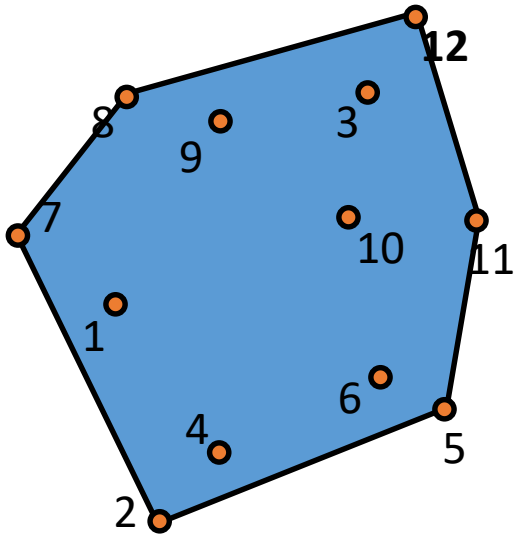
polygon.h

```
typedef struct
{
    int x, y
} Point;
typedef struct _Vertex
{
    struct _Vertex *next;
    Point p;
    int vis;
} Vertex;

Vertex* Create( Point p );
void InsertAfter( Vertex *v , Point p );
void DeleteAfter( Vertex *v );
```

onvex hull of points in 2D?

port dynamic insertion / deletion



Convex Hulls

polygon.h

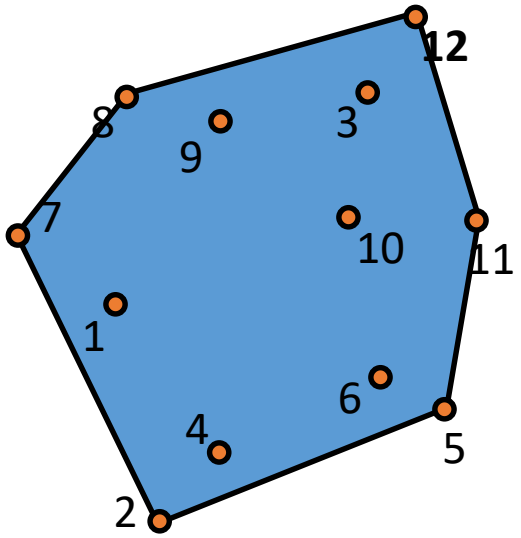
```
typedef struct
{
    int x, y
} Point;
typedef struct _Vertex
{
    struct _Vertex *next;
    Point p;
    int vis;
} Vertex;

Vertex* Create( Point p );
void InsertAfter( Vertex *v , Point p );
void DeleteAfter( Vertex *v );
```

polygon.c

```
#include <stdlib.h>
#include "polygon.h"
#include "assert.h"

Vertex* Create( Point p )
{
    Vertex *v = malloc( sizeof( Vertex ) );
    assert( v );
    v->next = v ; v->p = p;
    return v;
}
```



Convex Hulls

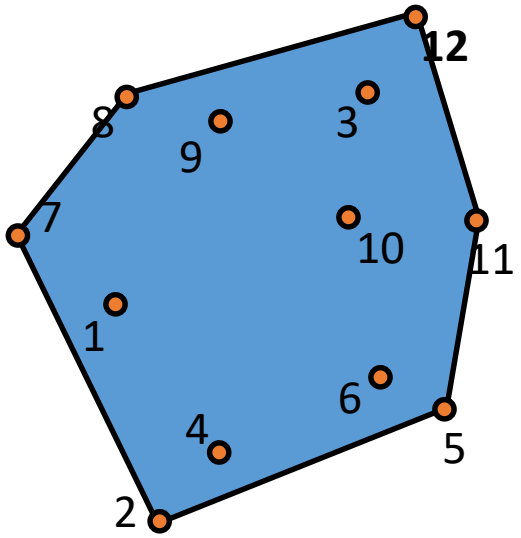
polygon.h

```
typedef struct
{
    int x, y
} Point;
typedef struct _Vertex
{
    struct _Vertex *next;
    Point p;
    int vis;
} Vertex;

Vertex* Create( Point p );
void InsertAfter( Vertex *v , Point p );
void DeleteAfter( Vertex *v );
```

polygon.c

```
#include <stdlib.h>
#include "polygon.h"
#include "assert.h"
...
void InsertAfter( Vertex* v , Point p )
{
    Vertex* newV = Create( p );
    newV->next = v->next;
    v->next = newV;
}
```



Convex Hulls

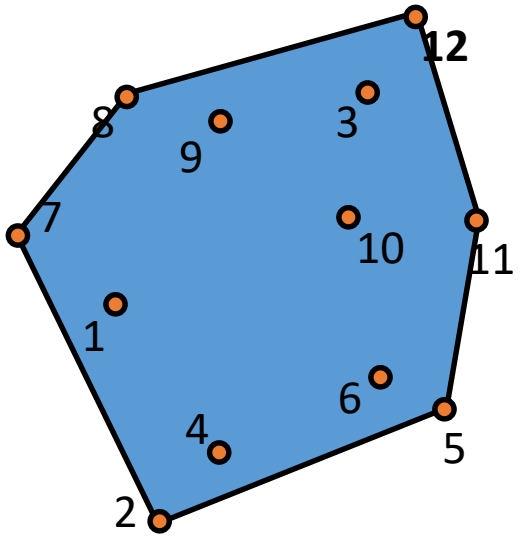
polygon.h

```
typedef struct
{
    int x, y
} Point;
typedef struct _Vertex
{
    struct _Vertex *next;
    Point p;
    int vis;
} Vertex;

Vertex* Create( Point p );
void InsertAfter( Vertex *v , Point p );
void DeleteAfter( Vertex *v );
```

polygon.c

```
#include <stdlib.h>
#include "polygon.h"
#include "assert.h"
...
void DeleteAfter( Vertex* v )
{
    Vertex* vNext = v->next;
    v->next = vNext->next;
    free( vNext );
}
```



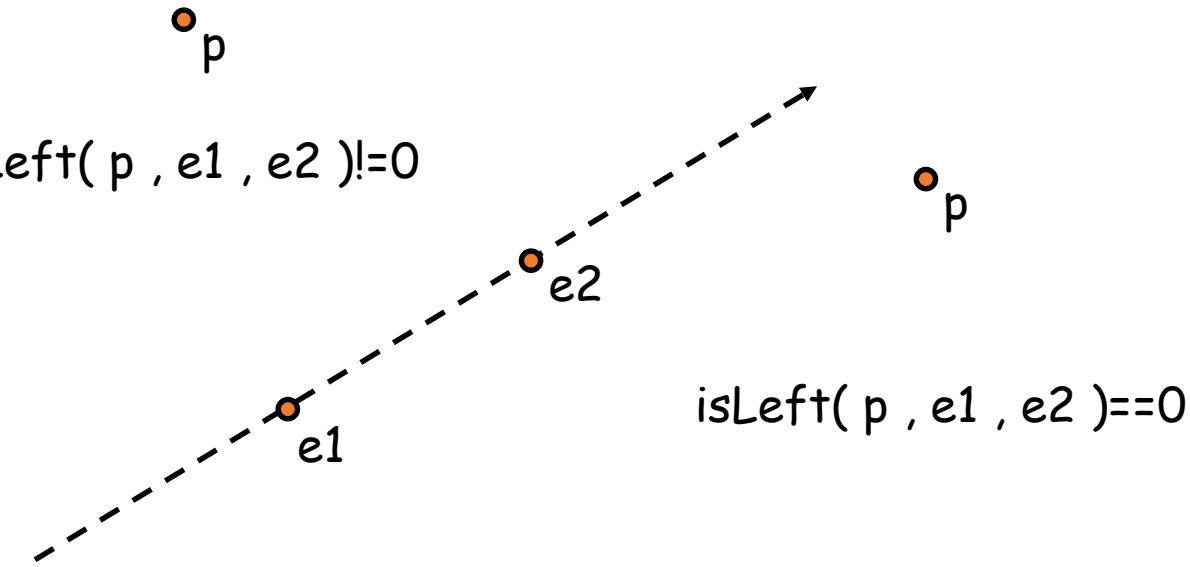
Convex Hulls

convexhull.c

```
#include <stdlib.h>
#include "polygon.h"

int isLeftE( Point p , Point e1 , Point e2 );
```

- Assume that we are given a function that checks if a point is to the left of the edge going from **e1** to **e2**



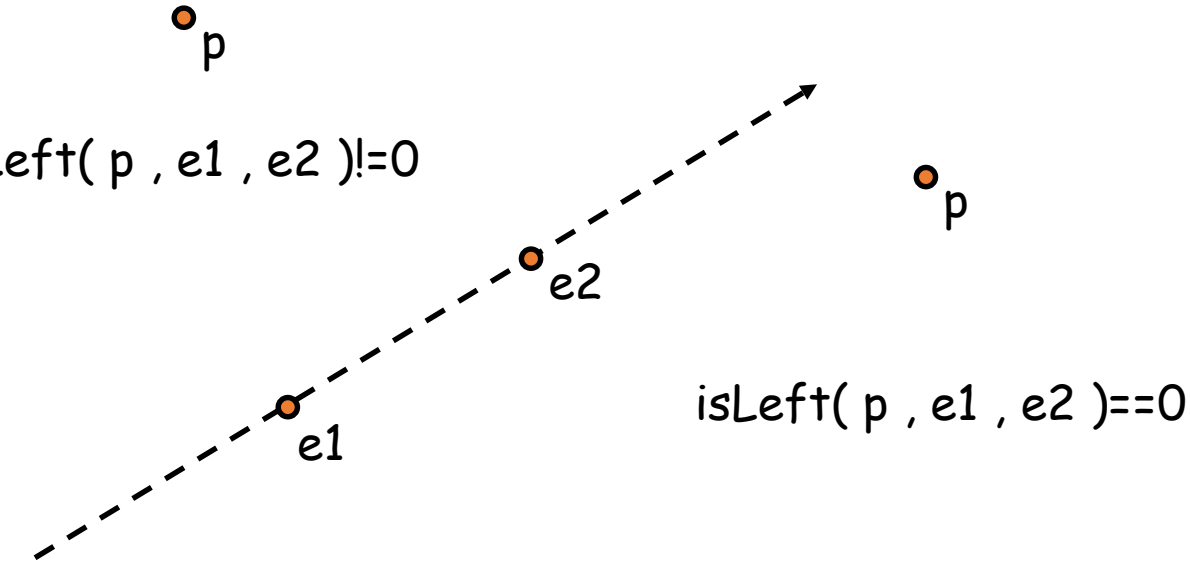
Convex Hulls

- Assume that we are given a function that checks if a point is to the left of the edge going from *e1* to *e2*
- We can extend this to a function that checks if a point is to the left of the edge of a polygon

convexhull.c

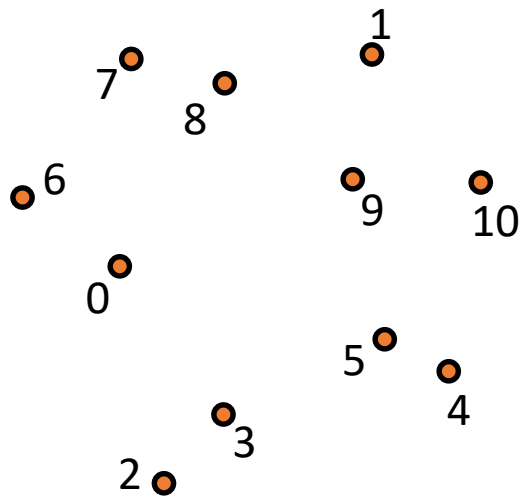
```
#include <stdlib.h>
#include "polygon.h"

int isLeftE( Point p , Point e1 , Point e2 );
int isLeftV( Point p , const Vertex* v )
{
    return isLeftE( p , v->p , v->next->p );
}
```



Convex Hulls

1. Construct the (convex) triangle through the first three points
2. Iterate over the remaining points and grow the convex hull to include them



convexhull.c

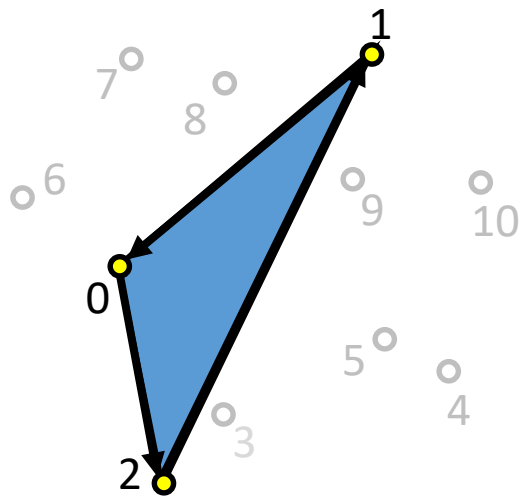
```
#include <stdlib.h>
#include "polygon.h"

Vertex* init( const Point* p );
void increment( Vertex* ch , Point p )

Vertex* hull( const Point* pts , int num )
{
    assert( num>=3 );
    Vertex* ch = init( pts );
    for( int i=3 ; i<num ; i++ )
        increment( ch , pts[i] );
    return ch;
}
```

Convex Hulls

1. **Construct the (convex) triangle through the first three points**
2. Iterate over the remaining points and grow the convex hull to include them



convexhull.c

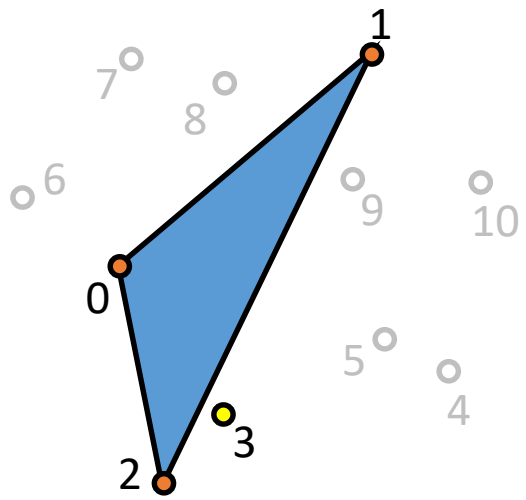
```
#include <stdlib.h>
#include "polygon.h"
…
Vertex* init( const Point* p )
{
    Vertex* tri = NULL;
    if( isLeftE( p[2] , p[0] , p[1] ) )
    {
        tri = Create( p[0] );
        InsertAfter( tri , p[1] );
        InsertAfter( tri->next , p[2] );
    }
    else
    {
        ch = Create( p[0] );
        InsertAfter( tri , p[1] );
        InsertAfter( tri , p[2] );
    }
    return tri;
}
```

Convex Hulls

1. Construct the (convex) triangle through the first three points
2. **Iterate over the remaining points and grow the convex hull to include them**

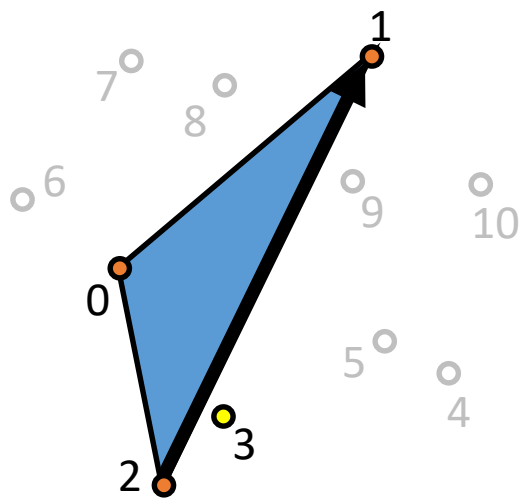
convexhull.c

```
#include <stdlib.h>
#include "polygon.h"
...
void increment( Vertex* ch , Point p )
{
      ...
}
```



Convex Hulls

1. Construct the (convex) triangle through the first three points
2. Iterate over the remaining points and grow the convex hull to include them
   A. **Mark right (visible) hull edges**

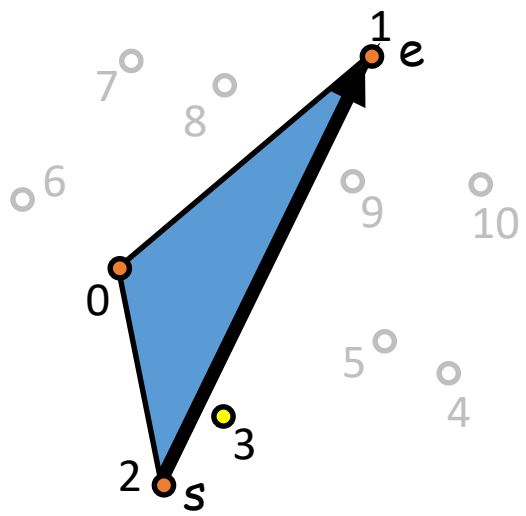


convexhull.c

```
#include <stdlib.h>
#include "polygon.h"
...
void increment( Vertex* ch , Point p )
{
    if( !markEdges( ch , p ) ) return;
}
int markEdges( Vertex* ch , Point p )
{
    Vertex* v = ch;
    int exterior = 0;
    do
    {
        v->vis = !isLeftV( p , v );
        exterior |= v->vis;
        v = v->next;
    }
    while( v!=ch );
    return exterior;
}
```

Convex Hulls

1. Construct the (convex) triangle through the first three points
2. Iterate over the remaining points and grow the convex hull to include them
   A. Mark right (visible) hull edges
   B. Identify visible / invisible transition vertices

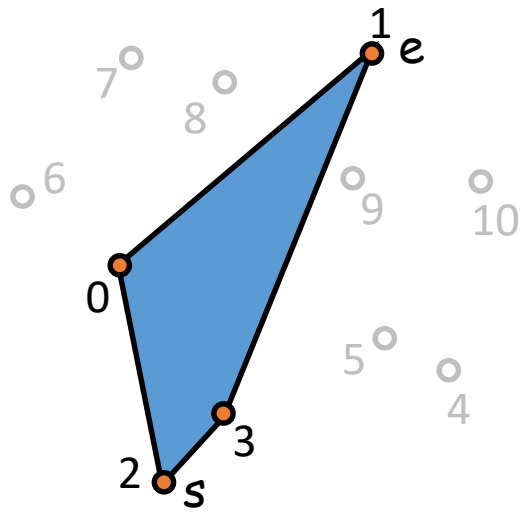


convexhull.c

```
#include <stdlib.h>
#include "polygon.h"
...
void increment( Vertex* ch , Point p )
{
    if( !markEdges( ch , p ) ) return;
    Vertex *v=ch , *s , *e;
    do
    {
        if( !v->vis && v->next->vis )
            s = v->next;
        if( v->vis && !v->next->vis )
            e = v->next;
        v = v->next;
    }
    while( v!=ch );
}
```

Convex Hulls

1. Construct the (convex) triangle through the first three points
2. Iterate over the remaining points and grow the convex hull to include them
   A. Mark right (visible) hull edges
   B. Identify visible / invisible transition vertices
   C. Remove visible vertices and insert the point

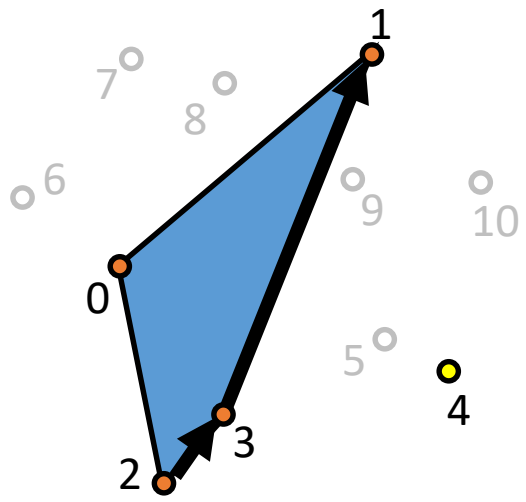


convexhull.c

```
#include <stdlib.h>
#include "polygon.h"
...
void increment( Vertex* ch , Point p )
{
    if( !markEdges( ch , p ) ) return;
    Vertex *v=ch , *s , *e;
    ...
    for( v=s ; v->next!=e ; v=v->next );
        DeleteAfter( v );
    InsertAfter( s , p );
}
```

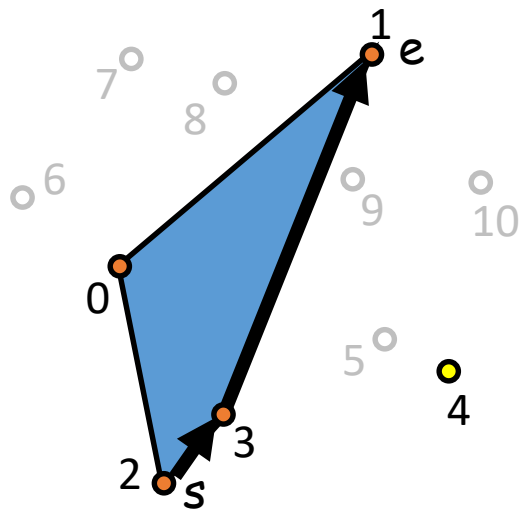
Convex Hulls

1. Construct the (convex) triangle through the first three points
2. Iterate over the remaining points and grow the convex hull to include them
   A. Mark right (visible) hull edges



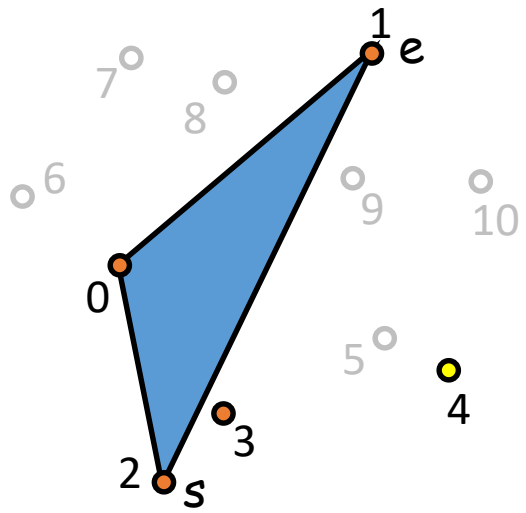
Convex Hulls

1. Construct the (convex) triangle through the first three points
2. Iterate over the remaining points and grow the convex hull to include them
   A. Mark right (visible) hull edges
   B. Identify visible / invisible transition vertices



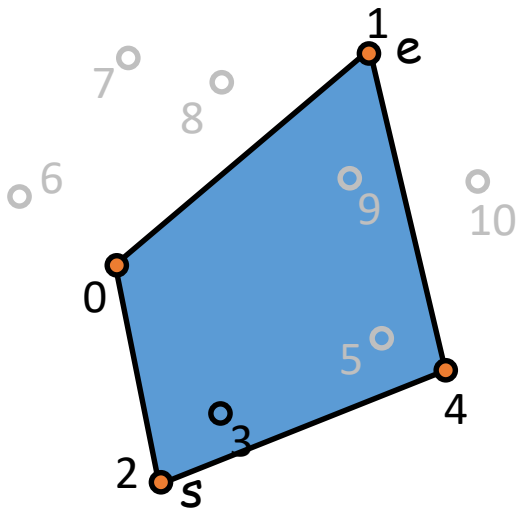
Convex Hulls

1. Construct the (convex) triangle through the first three points
2. Iterate over the remaining points and grow the convex hull to include them
   A. Mark right (visible) hull edges
   B. Identify visible / invisible transition vertices
   C. Remove visible vertices and insert the point



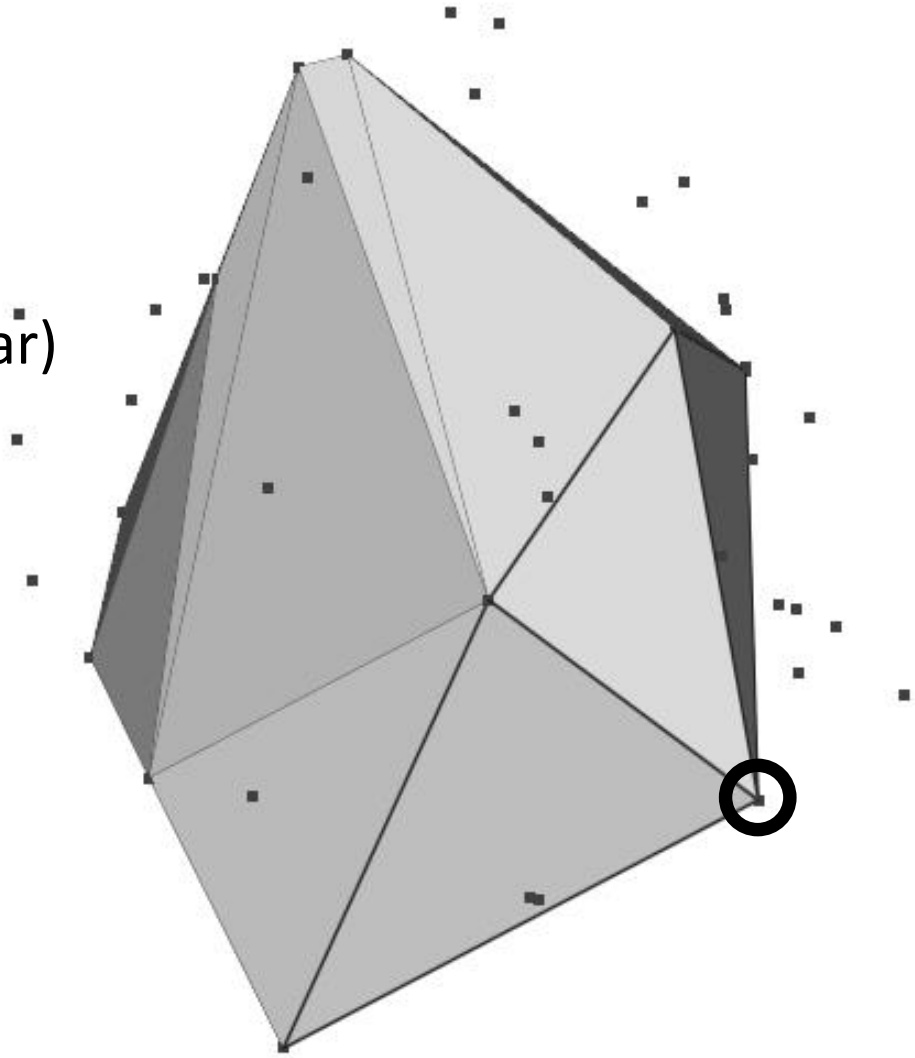
Convex Hulls

1. Construct the (convex) triangle through the first three points
2. Iterate over the remaining points and grow the convex hull to include them
   A. Mark right (visible) hull edges
   B. Identify visible / invisible transition vertices
   C. Remove visible vertices and insert the point



Convex Hulls

- The area of computational geometry studies these types of problems and likes to ask:
  - Are there faster algorithms? (Yes)
  - How fast can the fastest algorithm be? (Log-linear)
  - How does it behave if points are not in general position? (Stable but order-dependent)
  - Does it extend to higher dimensions? (Yes)



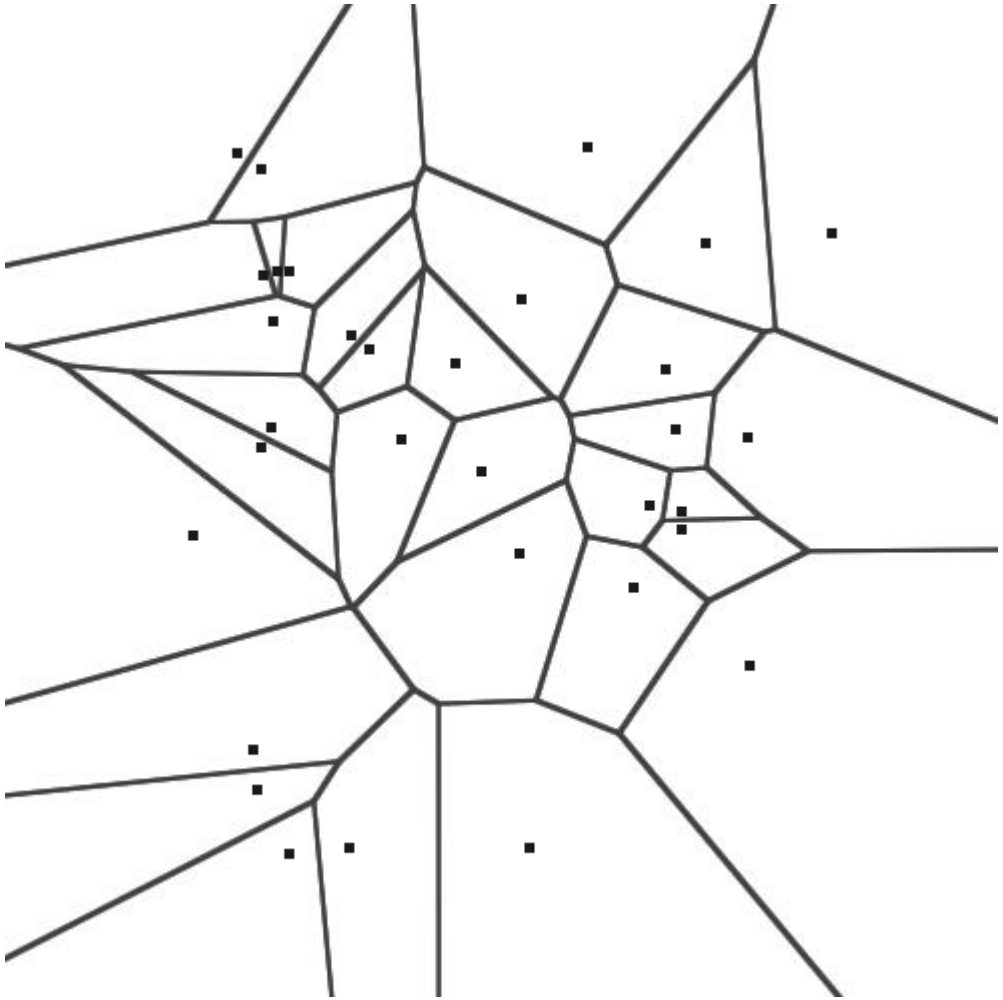
Convex Hulls

Q: Why do we care?

A: Suppose we have points in 2D, we would like to partition space into regions based on which point they are closest to

- E.g. points are cell-phone towers, network hubs, watch-towers in a forest, etc.

⇒ coverage



We can get the solution to this 2D problem by computing the convex hull in 3D!

Outline

- Convex hulls
- Midterm stuff

Midterm Format

- 75 minutes long, during class on Friday March 17th
- paper, closed book, closed notes, no computer or internet access
- possible types of questions:
  - short answer (multiple choice, fill-in-the-blank, write a short C code, etc.)
  - code tracing (given a C segment, predict its output)
  - longer code writing (provide a function or an entire program, including both main and functions).