GRAHAM_SCAN(Q) // $|Q| \geq 3$ insieme di punti sul piano

1. determina p_0, punto con ordinata minima // $|Q| = n$; se per p_i e p_j $i \neq j$ $y_{p_i} = y_{p_j}$ e $x_{p_i} < x_{p_j}$ => scelgo p_i $O(n)$

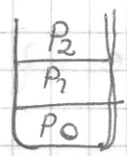
2. ordina $\{p_1, \ldots, p_n\} = Q \setminus \{p_0\}$ rispetto all'angolo polare formato rispetto a p_0; (in senso antiorario). // $O(n \log n)$

[rimuovere i punti allineati (uguale angolo)] // se p_i e p_j formano lo stesso angolo e p_j è più lontano di uno p_i, che si può ottenere per combinazione convessa $O(n)$

3. top(S) = 0 // funzione che mi restituisce il puntatore alla pila

4. push(p_0, S) //
5. push(p_1, S)
6. push(p_2, S)

} inizializzo la pila S con i primi 3 punti



top(S)

next top(S): restituisce il penultimo inserito

7. for i = 3 to n

8. do { while (l'angolo formato da next top(S), top(S), p_i non provoca una svolta a sx) // la verifica della svolta si effettua calcolando i prodotti incrociati $O(n)$ (*)

// inserisco p_i controllando di non creare una concavità -> elimino i precedenti che provocano con p_i svolta a dx

9. { pop(S) }

10. push(p_i, S) // poi inserisco p_i => successivamente può essere rimosso

}

11. return S

(*) il ciclo while annidato nel for può indurre una stima quadratica $O(n^2)$ della complessità dei passi 7. → 10

In realtà con il metodo degli aggregati per l'analisi ammortizzata si dimostra che questa fase costa $O(n)$

```
|Q| = n ;
P array ; // array dei punti ; dim (p) = n+1
swap P[1] con il punto di coord. y minimo ;
sort P rispetto all'angolo polare con P[1]
m = 2    // m mantiene il n° di punti in CH(Q)
p(0) = p(n) ;        // ci fermiamo quando troviamo il p0
for i = 3 to n    {
    while ( p(m-1), p(m), p(i) provocano
                  svolta a dx  )   {
        if (m = 2)                          // controllo di
                                            // collinearità
            { swap p(m) con p(i) ;          // --> ignoro pti
                                            //  non necessari
              i = i+1 ;                     }
        else { m = m-1 }
    }
    m = m+1 ;
    swap p(m) con p(i)
}
```