```
        }
        size_t operator()(uint64_t x) const { // x key
            return splitmix64(x);
        }
        size_t operator()(pair<uint64_t, uint64_t> x) const { // For, key = pair
            return splitmix64(x.first) ^ splitmix64(x.second);
        }
    };
```

- **priority_queue**<int>**max_heapPQ;** => In this queue elements are in non-increasing. [same as **multiset <int, greater<int> > s; But Priority Queue is more faster.**]
- **priority_queue**<int**,** vector<int>, **greater<int>>min_heapPQ;** => In this queue elements are in non-decreasing order. [similar to **multiset, but Priority Queue is more faster.**]].

Math:

- p+(p+1)+...+(q - 1)+q = **(q + p)(q – p + 1) / 2;** [Ex: **7**+8+9+10+**11**=(11+7)(11-7+1) / 2 =45]
- 1+2+3+...+(n - 1)+n = **(n * (n + 1)) / 2**; [Ex: 1+2+3+4+5=(5*(5+1))/2 = 15]
- 1+3+5+...+ (2n - 3) + (2n - 1) = **N^2**; [**N-> number of size**] [Ex: 1+3+5 = 3^2 = 9]
- 2+4+6+...+ (2n - 2) + 2n **= N * (N + 1)**; [**N-> number of size**] [Ex: 2+4+6 = 3*(3+1) =12]
- $1^2+2^2+3^2+...+(n-1)^2+n^2$ = **n (n + 1) (2n + 1) / 6;** [Ex: 1+4+9 = 3(3+1)(2*3 +1)/6 =14]
- $1^3+2^3+3^3+...+(n-1)^3+n^3$ = **$\{n(n+1)/2\}^2$;** [Ex: 1+8+27= $\{3(3+1)/2\}^2$ =36]
- $1^2+3^2+5^2+...+(2n-3)^2+(2n-1)^2$ = **$N*(4N^2-1)/3$;** [Ex: 1+9+25 = $3*(4*3^2-1)/3$ = 35]
- $1^3+3^3+5^3+...+(2n-3)^3+(2n-1)^3$ = **$N^2(2N^2-1)$;** [Ex: 1+27+125 = $3^2(2*3^2-1)$ = 153]
- $1^4+2^4+3^4+...+(n-1)^4+n^4$ = **$n(n+1)(2n+1)(3n^2+3n-1)/30$;**
  [Ex: 1+16+81+256 = $4(4+1)(2*4+1)(3*4^2+3*4-1)/30$ = 354]
- $c^a+c^{a+1}+\cdots+c^b$ = **$(c^{b+1}-c^a)/(c-1)$;** [c != 1]
- $2^0+2^1+2^2+2^3+...+2^{(k-1)}$ = **2^k-1;** [Ex: 1+2+4+8+16+32 = 2^6-1 = 63]
- If F(n) = -1 + 2 - 3 + . . .+ $(-1)^n$ * n
  - If N **even** number, **ans = N/2;**
  - If N **odd** number, **ans = ((N + 1) / 2) * (-1));**
- N-th **Odd** number = **(2 * N) – 1**;
- N-th **Even** number = **2*N**;
- a + a*k + a*k^2 + ... + b = **((b * k) - a) / (k - 1).** [ex: 3 + 6 + 12 + 24 = ((24 * 2) -3) / (2-1) = 45]
- a + (a+4) + (a+2*4) + ... + b= **(n * (a + b)) / 2.** [n-> number of size]
  [ex: 3 + 7 + 11 + 15 = (4 * (3 + 15)) / 2 = 36.]
- Number of digits in N = **floor(log10(**N**)) + 1;**
- Number of trailing **zeros** in **N!** => **while(N) sum+=N/5, N/=5; [**Ex: 10! = 36288**00;]**
- For a grid of size **(N x N)** the total number of **squares** formed: **((n*(n+1)) * (2n+1)) / 6;**
- **5 minutes** Clock Angular Value is **30°. [1 min = 6°]**
- Angle between clock minute and hour, **ans = abs ((0.5 * 11 * m) – (30 * h));**
  - For smaller angle, **if (ans >180) ans = 360 – ans;**
- The number of ways of selecting one or more things from N different things is given by
  2^N -1. (combination)
- Number of possible of N bits = 2^N. [4bits, 24 =16 => 0 to 15 number possible with using 4 bits] (2^n -1) → highest value.
- The number of possible **unique triplet** for an array of length **n** formula: **n * (n-1) * (n-2) / 6;**
- N = **2^x =>** x = **log2(N).** Ex: 64 = 2^6 [log2(64) = **6**].
- **$\log_u(x)$=** $\frac{logk(x)}{logk(u)}$ [k-> any base (2,10)]; **$\log_a(k)$=** $\frac{1}{logk(a)}$; **a^x = b ;=> x = $\log_a$ b;**
- **(A * B) = ((A % Mod) * (B % Mod)) % Mod;** **<=** [Same As **+,-** Operator]