SESSDSA'20

问题解答

MD5算法详解

【C2】Python开源硬件介绍

期末考试安排

数据结构与算法（Python）-14/0414

陈斌 gischen@pku.edu.cn 北京大学地球与空间科学学院

目录

问题解答

- 完美散列函数在处理大文件的时候是怎么做到快速计算的呢？
- 可以详细的讲一下Md5和SHA是通过什么样的hash算法生成的吗？
- 为什么散列值可以是字母呢，像256的散列ASCII代码不够，是直接用数字表示吗？
- 二次探测的skip值应该增加到多少呢？因为有可能返回到原位置后继续查找找到了空槽。

……二次探测是只要还冲突，就增加skip值，加到找到空槽为止

……当然另一个数冲突的skip还从1开始

问题解答

- OJ题用散列表反而不会写了，求助教给个散列表的答案。
  ……见OJ详解
- MOOC测验最后一题。
- 为啥H4最短编辑距离，sessdsa是最容易出错的输入？
- H4后预览显示的话是啥意思：“如果测评机给出了错误样例，首先在本地使用该输入运行自己提交的作业中的代码，并在批改结果有误或运行结果与在线不一致时联系助教（注：不包括报错信息不一致的情况)”

【H4】的SESSDSA用例


```
s,c=dpWordEdit("", "", oplist)
print(s,c)
s,c=dpWordEdit("", "SESSDSA", oplist)
print(s,c)
s,c=dpWordEdit("SESSDSA", "", oplist)
print(s,c)
```

› 有三个涉及到空串的用例

```
    # 逐个填写二维表
    i = j = 0
    for i in range(1, len(original)):
        for j in range(1, len(target)):
            ops = [] # 可能的操作和对应的分数
            # copy操作，有条件
            if original[i] == target[j]:
                ops.append([m[i-1, j-1][0] + oplist
            # delete操作
            ops.append([m[i-1,j][0] + oplist["delet
            # insert操作
            ops.append([m[i,j-1][0] + oplist["inser

            # 取分数最小的
            m[i,j] = min(ops, key=lambda x:x[0])

    if i == 0:
        j = len(target) - 1
    score = m[i,j][0]
    while i> 0 or j> 0:
```

【H4】博物馆大盗用例

数据结构与算法（Python）

- 有涉及到拿不满最大重量的情况

```
m,c=dpMuseumThief([{'w':1,'v':24},{'w':18,'v':8},{'w':1,'v':24}],4)
print(m,c)
```

```
    maxValue = m[i, w][0]
    while i > 0:
        if m[i, w][1] is not None: # 如果装了宝物，就输出
            chosenList.insert(0, m[i, w][1])
            w = w - m[i, w][1]['w']
        i = i - 1
```

问题解答

- **请问还有micro:bit吗？**

  ……【C2】

- **大作业规则定了吗？能否讲解一下大作业的代码？**

  ……F19/N19分区之间无竞争，原则上5人一组。将在5月中旬公布。

- **期末测试方式有变吗？会改成pf吗？**

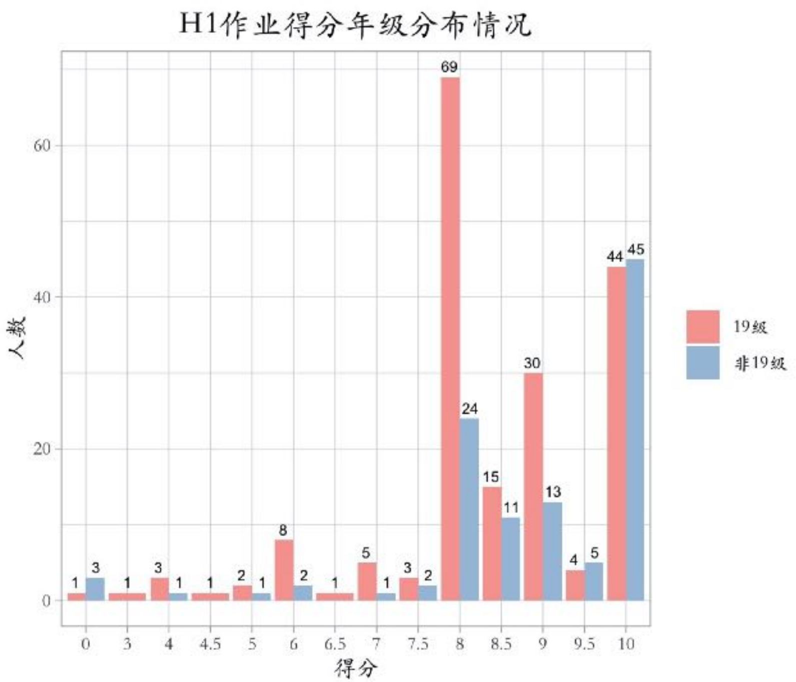
  ……NO PF

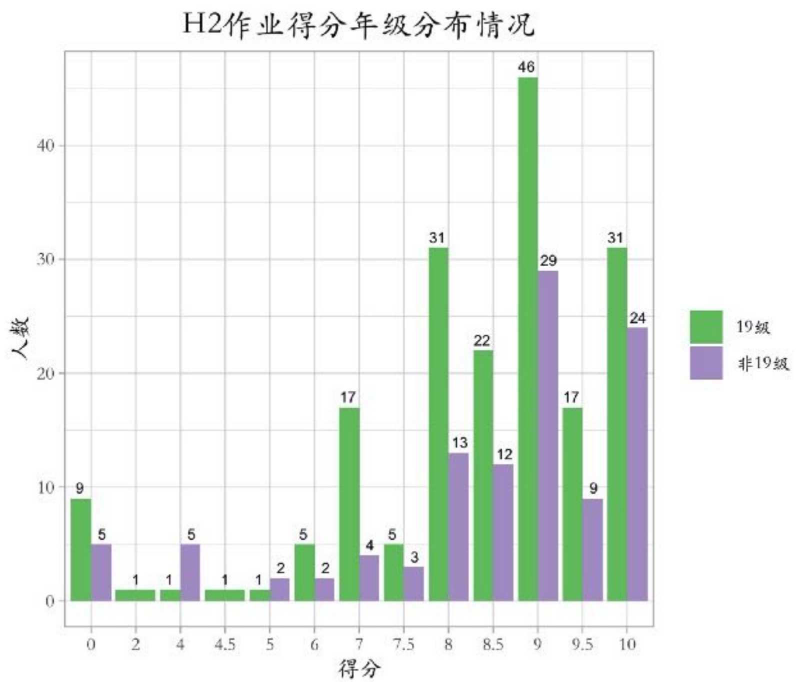
- **线上期末考试可以分摊到平时作业和大作业和出勤上吗？**

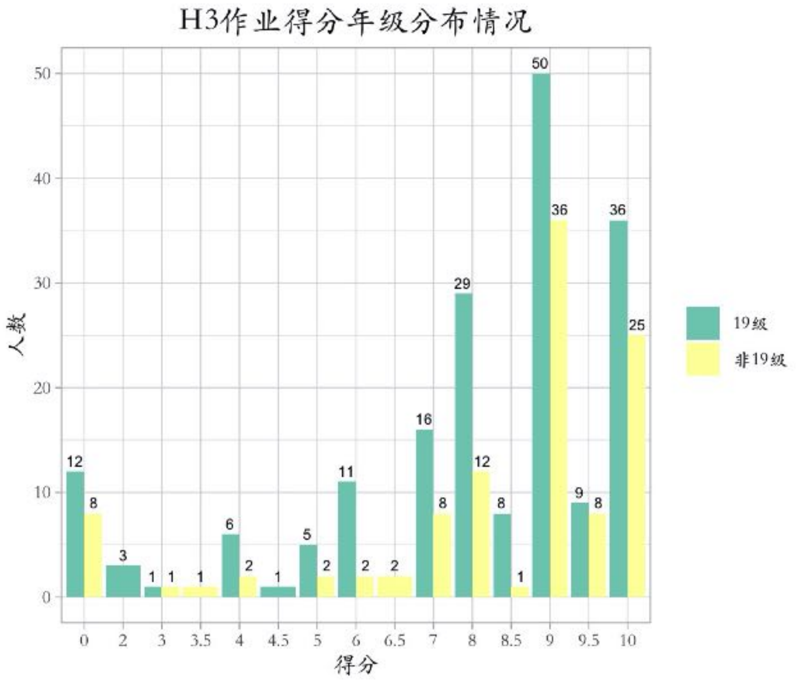
  ……如果本学期未能返校进行线下闭卷考试，则分摊。

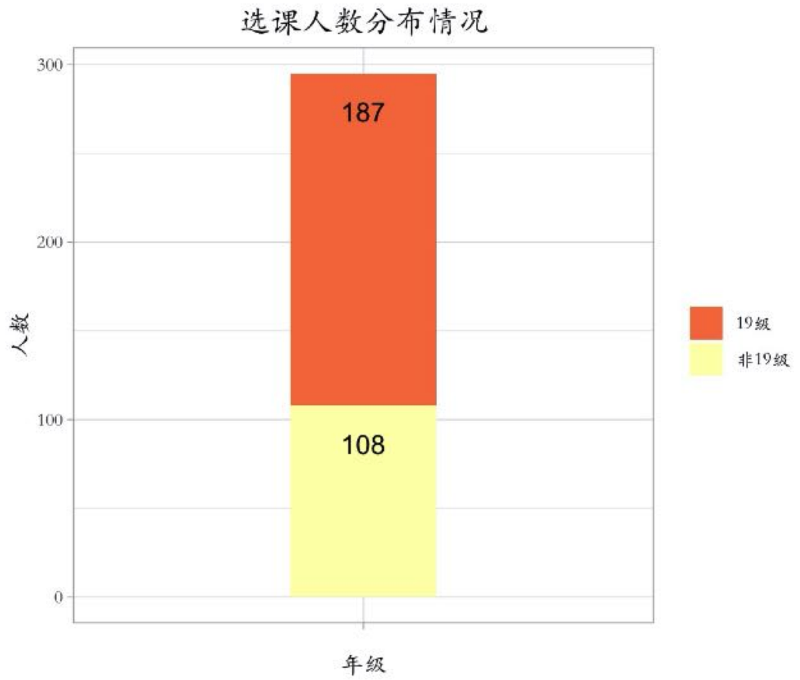
- **可以在退课结束之前看到自己的平时分成绩吗？对自己的平时分很没有把握。**

数据结构与算法（Python）









MD5算法

- MD5的全称是Message-Digest Algorithm
- MD5由美国密码学家罗纳德·李维斯特（Ronald Linn Rivest）设计，于1992年公开，用以取代MD4算法。

  RFC-1321给出了标准规范的MD5算法
- MD5是输入不定长度信息，输出固定长度128-bits的算法。
- 经过程序流程，生成四个32位数据，最后联合起来成为一个128-bits散列。

  基本方式为，求余、取余、调整长度、与链接变量进行循环运算。
- https://zh.wikipedia.org/wiki/MD5

MD5算法

- MD5以512位分组来处理输入的信息，且每一分组又被划分为16个32位子分组
- 经过了一系列的处理后，算法的输出由四个32位分组组成，将这四个32位分组级联后将生成一个128位散列值。
- 填充：将10....加到明文数据后面，让长度凑到L%512=448，再添加64bits表示明文长度，这样就能512bits分组

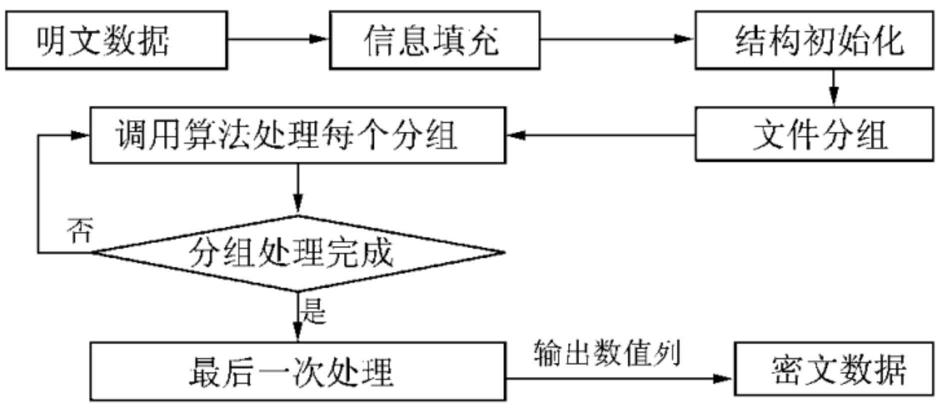


图 1 MD5算法流程图

MD5算法伪代码


```
//Note: All variables are unsigned 32 bits and wrap modulo 2^32 when calculating
var int[64] r, k

//r specifies the per-round shift amounts
r[ 0..15]: = {7, 12, 17, 22,  7, 12, 17, 22,  7, 12, 17, 22,  7, 12, 17, 22}
r[16..31]: = {5,  9, 14, 20,  5,  9, 14, 20,  5,  9, 14, 20,  5,  9, 14, 20}
r[32..47]: = {4, 11, 16, 23,  4, 11, 16, 23,  4, 11, 16, 23,  4, 11, 16, 23}
r[48..63]: = {6, 10, 15, 21,  6, 10, 15, 21,  6, 10, 15, 21,  6, 10, 15, 21}

//Use binary integer part of the sines of integers as constants:
for i from 0 to 63
    k[i] := floor(abs(sin(i + 1)) × 2^32)

//Initialize variables:
var int h0 := 0x67452301
var int h1 := 0xEFCDAB89
var int h2 := 0x98BADCFE
var int h3 := 0x10325476

//Pre-processing:
append "1" bit to message
append "0" bits until message length in bits ≡ 448 (mod 512)
append bit length of message as 64-bit little-endian integer to message
```

```
//Process the message in successive 512-bit chunks:
for each 512-bit chunk of message
    break chunk into sixteen 32-bit little-endian words w[i], 0 ≤ i ≤ 15

    //Initialize hash value for this chunk:
    var int a := h0
    var int b := h1
    var int c := h2
    var int d := h3

    //Main loop:
    for i from 0 to 63
        if 0 ≤ i ≤ 15 then
            f := (b and c) or ((not b) and d)
            g := i
        else if 16 ≤ i ≤ 31
            f := (d and b) or ((not d) and c)
            g := (5×i + 1) mod 16
        else if 32 ≤ i ≤ 47
            f := b xor c xor d
            g := (3×i + 5) mod 16
        else if 48 ≤ i ≤ 63
            f := c xor (b or (not d))
            g := (7×i) mod 16

        temp := d
        d := c
        c := b
        b := leftrotate((a + f + k[i] + w[g]),r[i]
        a := temp
    Next i
    //Add this chunk's hash to result so far:
    h0 := h0 + a
    h1 := h1 + b
    h2 := h2 + c
    h3 := h3 + d
End ForEach
var int digest := h0 append h1 append h2 append h3 //(expressed as little-endian)
```


```
//Main loop:
for i from 0 to 63
    if 0 ≤ i ≤ 15 then
        f := (b and c) or ((not b) and d)
        g := i
    else if 16 ≤ i ≤ 31
        f := (d and b) or ((not d) and c)
        g := (5×i + 1) mod 16
    else if 32 ≤ i ≤ 47
        f := b xor c xor d
        g := (3×i + 5) mod 16
    else if 48 ≤ i ≤ 63
        f := c xor (b or (not d))
        g := (7×i) mod 16

    temp := d
    d := c
    c := b
    b := leftrotate((a + f + k[i] + w[g]),r[i]) + b
    a := temp
Next i
```

计算一个文件的MD5值


```
import hashlib

blk_size = 2048  # 每次读取的字节数
m = hashlib.md5() # 创建一个散列函数对象
with open("2004-199.pdf", "rb") as f:
    while True:
        buf = f.read(blk_size)
        if not buf:
            break
        m.update(buf)  # 更新

print(m.hexdigest())  # 打印散列值的16进制文本串
```

```
>>> %Run hashtest.py
  7667d184375a8d968e9e107217f7e8ea
```

MD散列冲突构造

Collisions for Hash Functions

MD4, MD5, HAVAL-128 and RIPEMD

Xiaoyun Wang[1], Dengguo Feng[2], Xuejia Lai[3], Hongbo Yu[1]

The School of Mathematics and System Science, Shandong University, Jinan250100, China[1]

Institute of Software, Chinese Academy of Sciences, Beijing100080, China[2]

Dept. of Computer Science and Engineering, Shanghai Jiaotong University, Shanghai, China[3]

xywang@sdu.edu.cn[1]

revised on August 17, 2004

1 Collisions for MD5

MD5 is the hash function designed by Ron Rivest [9] as a strengthened version of MD4 [8]. In 1993 Bert den Boer and Antoon Bosselaers [1] found pseudo-collision for MD5 which is made of the same message with two different sets of initial value. H. Dobbertin[3] found a free-start collision which consists of two different 512-bit messages with a chosen initial value IV_0'.

$$IV_0': A_0' = 0x12AC2375,\ B_0' = 0x3B341042, C_0' = 0x5F62B97C, D_0' = 0x4BA763ED$$

Our attack can find many real collisions which are composed of two 1024-bit messages with the original initial value IV_0 of MD5:

https://blog.csdn.net/SysProgram/article/details/73753354

SHA家族

2008 AMD CPU

SHA函数对比

算法和变体		输出散列值长度 (bits)	中继散列值长度 (bits)	数据区块长度 (bits)	最大输入消息长度 (bits)	循环次数	使用到的运算符	碰撞攻击 (bits)	性能示例[3] (MiB/s)
MD5（作为参考）		128	128 (4 × 32)	512	无限[4]	64	And, Xor, Rot, Add (mod 2^{32}), Or	≤18 (发现碰撞)	335
SHA-0		160	160 (5 × 32)	512	$2^{64} - 1$	80	And, Xor, Rot, Add (mod 2^{32}), Or	<34 (发现碰撞)	–
SHA-1		160	160 (5 × 32)	512	$2^{64} - 1$	80		<63[5] (发现碰撞[6])	192
SHA-2	*SHA-224* *SHA-256*	224 256	256 (8 × 32)	512	$2^{64} - 1$	64	And, Xor, Rot, Add (mod 2^{32}), Or, Shr	112 128	139
	SHA-384 *SHA-512* *SHA-512/224* *SHA-512/256*	384 512 224 256	512 (8 × 64)	1024	$2^{128} - 1$	80	And, Xor, Rot, Add (mod 2^{64}), Or, Shr	192 256 112 128	154
SHA-3	*SHA3-224* *SHA3-256* *SHA3-384* *SHA3-512*	224 256 384 512	1600 (5 × 5 × 64)	1152 1088 832 576	无限[7]	24[8]	And, Xor, Rot, Not	112 128 192 256	–
	SHAKE128 *SHAKE256*	*d* (arbitrary) *d* (arbitrary)		1344 1088				min(d/2, 128) min(d/2, 256)	–

慕课测验详解

- OJ作业详解见Canvas/gis4g上的助教PDF

【C2】创意作品：开源硬件

活动目标

- 练习Python编程；
- 体验性能受限环境中的算法优化；
- 编程实现算法驱动的智能硬件作品。

开发平台

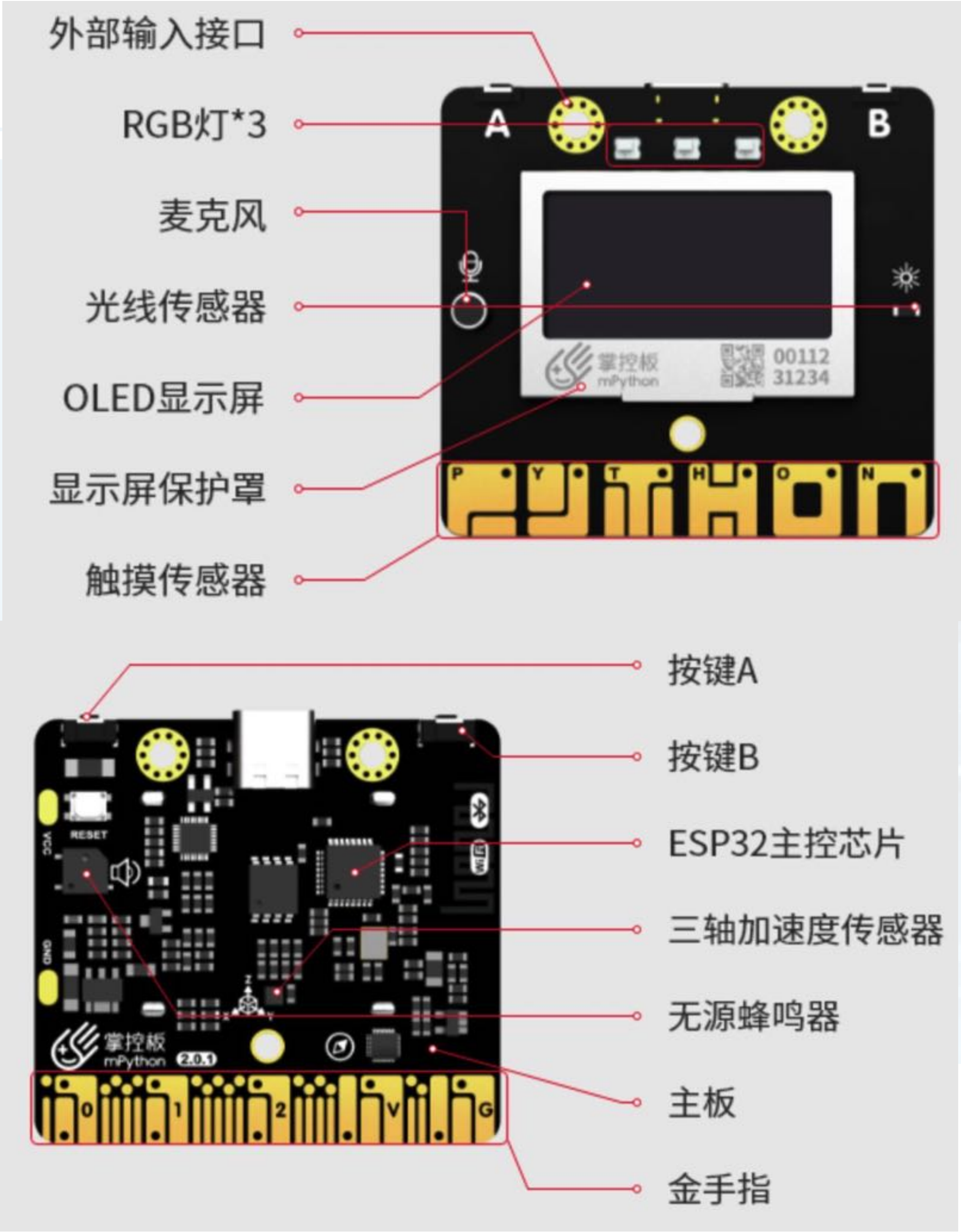
- 核心板1：掌控板2.0：https://www.mpython.cn/mPython/index
- 核心板2：Sipeed MAIX Dock：https://maixpy.sipeed.com
- 要求作品物料（包括核心板）总成本在RMB200以内。

活动要求

- 本课同学自愿参与，每组限2人；
- 第一阶段，在周六（4月18日）前，通过阅读文档，咨询厂家，熟悉开发平台，提出**项目建议**；
  - 经老师和助教评审后，根据课程活动经费预算，**最多选出30组予以资助**，作品完成后每组可获补助RMB200；
- 第二阶段，受资助的组进行作品创作，5月10日前提交作品（源码、技术报告、2min视频）；
  - 也欢迎未受资助的组参与；
  - 提交作品要求可参考：http://gis4g.pku.edu.cn/microbit-artwork-2018/

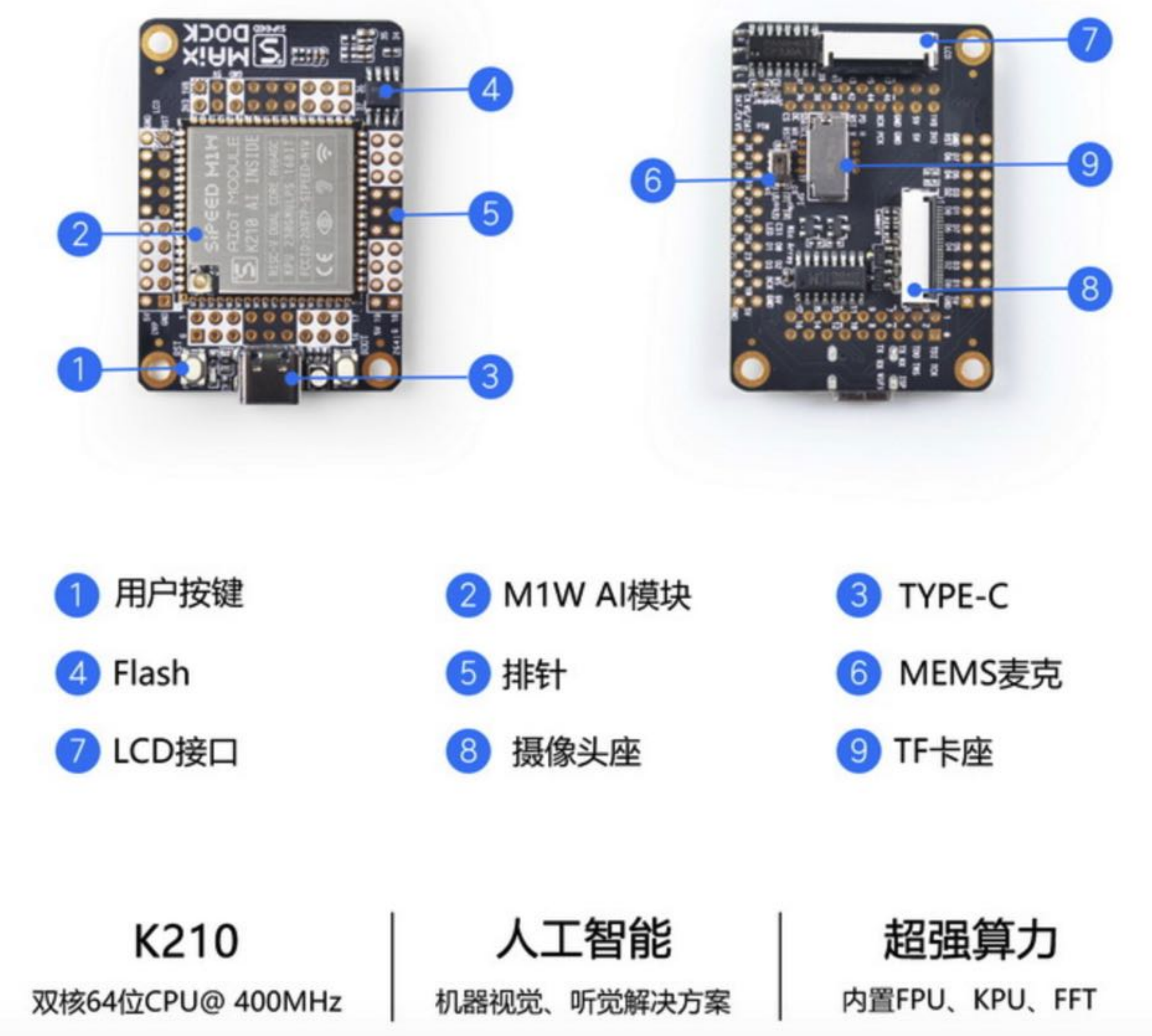
开发板1：掌控板2.0

- **ESP32主控芯片**
  - 双核处理器，240MHz
  - 520K RAM/8MB Flash
  - WiFi/蓝牙
  - 内置OLED显示屏
  - 内置各种传感器



开发板2：MAIX Dock

- **国产RISC V芯片K210**
  - 双核64bit，400MHz
  - 8MB RAM/16M Flash
  - WiFi
  - 摄像头/LCD接口
  - CNN神经网络加速
  - 图像、视频、音频处理



关于期末考试

- **本课：评分方式（可能会微调）**

  线上慕课成绩（25%）：包括W系列测验，必做的OJ作业，慕课期末考试

  线下小测和作业（25%）：包括K、H系列作业，每周二的签到

  分组实习大作业（20%）

  闭卷期末考试（30%）

- **关于闭卷期末考试**

  如果本学期能够返校进行线下闭卷期末考试，则不变；

  否则，初步变更为：

  期末报告15%，实习大作业25%，线下小测和作业35%，线上慕课25%