پاسخ سؤال ۹:

الف)
الگوریتم Heap Sort به حافظه اضافی نیاز ندارد (درجا انجام می‌شود)، چون مرتب‌سازی را در آرایه اصلی انجام می‌دهد، نه آرایه‌ای جدید.

ب)
Heap Sort الگوریتم Stable نیست، زیرا در حین Heapify، ممکن است ترتیب نسبی عناصر مساوی تغییر کند.

پ)
خیر، Heap Sort برای داده‌های تقریباً مرتب شده انتخاب خوبی نیست. الگوریتم‌هایی مانند Insertion Sort برای داده‌های تقریباً مرتب بهتر هستند، چون در این شرایط عملکرد آن‌ها نزدیک به خطی است.

ت)
برای ساخت Max Heap از یک آرایه می‌توان از الگوریتم Heapify استفاده کرد که پیچیدگی زمانی آن O(n) است، نه O(n log n). این تحلیل از طریق محاسبه تعداد سطوح درخت و مجموع عملیات در هر سطح به‌دست می‌آید.

ث)
در بدترین حالت، زمان لازم برای حذف عنصر از Heap برابر O(log n) است، چون باید عنصر جایگزین شده را به موقعیت مناسبش منتقل کنیم.
پاسخ درست: گزینه (a) یعنی O(log n)

پاسخ سؤال ۱۱:

ساخت Max Heap و Min Heap از آرایه داده شده:
arr = [10, 20, 35, 2, 14, 23, 70, 45, 56]

مرحله ۱: ساخت Max Heap
Max Heap باید خاصیت داشته باشد که هر پدر از فرزندانش بزرگ‌تر باشد.

درخت Max Heap (ساخت دستی یا با الگوریتم heapify):

```
        70
      /    \
```

مرحله ۲: ساخت Min Heap
Min Heap باید خاصیت داشته باشد که هر پدر از فرزندانش کوچک‌تر باشد.

درخت Min Heap:

لیست نهایی (array form):

* Max Heap: [70, 45, 35, 20, 56, 23, 10, 2, 14]
* Min Heap: [2, 10, 23, 14, 20, 70, 35, 45, 56]

پاسخ سؤال ۱۳: طراحی سیستم اولویت:

برای طراحی یک سیستم صف اولویت که بیماران را بر اساس شدت وضعیتشان مدیریت کند:

الویت‌ها:

1. بیماران با شدت بالا (عددی کمتر) اولویت بالاتری دارند.
2. در صورت مساوی بودن شدت، کسی که زودتر مراجعه کرده اولویت دارد (بر اساس زمان ورود).

ساختار مناسب:

* استفاده از **Min Heap** با اولویت‌های دوگانه: (شدت بیماری، زمان ورود)
* عناصر Heap به صورت تاپل ذخیره می‌شوند، مثلا: (2, timestamp)

**پاسخ سؤال

```
import heapq

class MyMinHeap:
    def __init__(self):
        self.heap = []

    def insert(self, value):
        heapq.heappush(self.heap, value)

    def get_min(self):
        return self.heap[0] if self.heap else None

# مثال استفاده:
h = MyMinHeap()
h.insert(10)
h.insert(5)
h.insert(20)
print(h.get_min())  # خروجی: 5
```
